

Revisiting Robustness in Priced Timed Games

S. Guha¹, S. N. Krishna², L. Manasa², and A. Trivedi²

1 Department of Computer Science & Engineering, IIT Delhi, India.
shibashis@cse.iitd.ernet.in

2 Department of Computer Science & Engineering, IIT Bombay, India.
krishnas,manasa,trivedi@cse.iitb.ac.in

Abstract

Priced timed games are optimal-cost reachability games played between two players—the controller and the environment—by moving a token along the edges of infinite graphs of configurations of priced timed automata. The goal of the controller is to reach a given set of target locations as cheaply as possible, while the goal of the environment is the opposite. Priced timed games are known to be undecidable for timed automata with 3 or more clocks, while they are known to be decidable for automata with 1 clock. In an attempt to recover decidability for priced timed games Bouyer, Markey, and Sankur studied robust priced timed games where the environment has the power to slightly perturb delays proposed by the controller. Unfortunately, however, they showed that the natural problem of deciding the existence of optimal limit-strategy—optimal strategy of the controller where the perturbations tend to vanish in the limit—is undecidable with 10 or more clocks. In this paper we revisit this problem and improve our understanding of the decidability of these games. We show that the limit-strategy problem is already undecidable for a subclass of robust priced timed games with 5 or more clocks. On a positive side, we show the decidability of the existence of almost optimal strategies for the same subclass of one-clock robust priced timed games by adapting a classical construction by Bouyer et al. for one-clock priced timed games.



© Guha, Krishna, Manasa and Trivedi;

licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Two-player zero-sum games on priced timed automata provide a mathematically elegant modeling framework for the control-program synthesis problem in real-time systems. In these games, two players—the *controller* and the *environment*—move a token along the edges of the infinite graph of configurations of a timed automaton to construct an infinite execution of the automaton in order to optimize a given performance criterion. The optimal strategy of the controller in such game then corresponds to control-program with the optimal performance. By priced timed games (PTGs) we refer to such games on priced timed automata with optimal reachability-cost objective. The problem of deciding the existence of the optimal controller strategy in PTGs is undecidable [8] with 3 or more clocks, while it is known to be decidable [5] for automata with 1 clock. Also, the ε -optimal strategies can be computed for priced timed games under the non-Zeno assumption [1, 4]. Unfortunately, however, the optimal controller strategies obtained as a result of solving games on timed automata may not be physically realizable due to unrealistic assumptions made in the modeling using timed automata, regarding the capability of the controller in enforcing precise delays. This severely limits the application of priced timed games in control-program synthesis for real-time systems.

In order to overcome this limitation, Bouyer, Markey, and Sankur [7] argued the need for considering the existence of robust optimal strategies and introduced two different robustness semantics—*excess* and *conservative*—in priced timed games. The key assumption in their modeling is that the controller may not be able to apply an action at the exact time delays suggested by the optimal strategy. This phenomenon is modeled as a *perturbation* game where the time delay suggested by the controller can be perturbed by a bounded quantity. Notice that such a perturbation may result in the guard of the corresponding action being disabled. In the conservative semantics, it is the controller’s responsibility to make sure that the guards are satisfied after the perturbation. On the other hand, in the excess semantics, the controller is supposed to make sure that the guard is satisfied before the perturbation: an action can be executed even when its guard is disabled (“excess”) post perturbation and the valuations post perturbation will be reflected in the next state. The game based characterization for robustness in timed automata under “excess” semantics was first proposed by Bouyer, Markey, and Sankur [6] where they study the parameterized robust (qualitative) reachability problem and show it to be EXPTIME-complete. The “conservative” semantics were studied for reachability and Büchi objectives in [13] and shown to be PSPACE-complete. For a detailed survey on robustness in timed setting we refer to an excellent survey by Markey [11].

Bouyer, Markey, and Sankur [7] showed that the problem for deciding the existence of the optimal strategy is undecidable for priced timed games with 10 or more clocks under the excess semantics. In this paper we further improve the understanding of the decidability of these games. However, to keep the presentation simple, we restrict our attention to turn-based games under excess semantics. To further generalize the setting, we permit both positive and negative price rates with the restriction that the accumulated cost in any cycle is non-negative (akin to the standard no-negative-cycle restriction in shortest path game problems on finite graphs). We improve the undecidability result of [7] by proving that optimal reachability remains undecidable for robust priced timed automata with 5 clocks. Our second key result is that, for a fixed δ , the cost optimal reachability problem for one clock priced timed games with no-negative-cycle restriction is decidable for robust priced timed games with given bound on perturbations. To the best of our knowledge, this is the first decidability result known for robust timed games under the excess semantics. A closely related result is [9], where decidability is shown for robust timed games under the conservative semantics for a fixed δ .

2 Preliminaries

We write \mathbb{R} for the set of reals and \mathbb{Z} for the set of integers. Let \mathcal{C} be a finite set of real-valued variables called *clocks*. A *valuation* on \mathcal{C} is a function $\nu : \mathcal{C} \rightarrow \mathbb{R}$. We assume an arbitrary

but fixed ordering on the clocks and write x_i for the clock with order i . This allows us to treat a valuation ν as a point $(\nu(x_1), \nu(x_2), \dots, \nu(x_n)) \in \mathbb{R}^{|\mathcal{C}|}$. Abusing notations slightly, we use a valuation on \mathcal{C} and a point in $\mathbb{R}^{|\mathcal{C}|}$ interchangeably. For a subset of clocks $X \subseteq \mathcal{C}$ and valuation $\nu \in \mathbb{R}^{|\mathcal{C}|}$, we write $\nu[X:=0]$ for the valuation where $\nu[X:=0](x) = 0$ if $x \in X$, and $\nu[X:=0](x) = \nu(x)$ otherwise. The valuation $\mathbf{0} \in \mathbb{R}^{|\mathcal{C}|}$ is a special valuation such that $\mathbf{0}(x) = 0$ for all $x \in \mathcal{C}$. A clock constraint over \mathcal{C} is a subset of $\mathbb{R}^{|\mathcal{C}|}$. We say that a constraint is *rectangular* if it is a conjunction of a finite set of constraints of the form $x \bowtie k$, where $k \in \mathbb{Z}$, $x \in \mathcal{C}$, and $\bowtie \in \{<, \leq, =, >, \geq\}$. For a constraint $g \in \varphi(\mathcal{C})$, we write $\llbracket g \rrbracket$ for the set of valuations in $\mathbb{R}^{|\mathcal{C}|}$ satisfying g . We write $\varphi(\mathcal{C})$ for the set of rectangular constraints over \mathcal{C} . We use the terms constraints and guards interchangeably.

Following [5] we introduce priced timed games with external cost function on target locations (see Appendix A). For this purpose, we define a *cost function* [5] as a piecewise affine continuous function $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R} \cup \{+\infty, -\infty\}$. We write \mathcal{F} for the set of all cost functions.

► **Definition 1 (Priced Timed Games).** A turn-based two player *priced timed game* is a tuple $\mathcal{G} = (L_1, L_2, L_{init}, \mathcal{C}, X, \eta, T, f_{goal})$ where L_i is a finite set of *locations* of Player i , $L_{init} \subseteq L_1 \cup L_2$ (let $L_1 \cup L_2 = L$) is a set of initial locations, \mathcal{C} is an (ordered) set of *clocks*, $X \subseteq L \times \varphi(\mathcal{C}) \times 2^{\mathcal{C}} \times (L \cup T)$ is the *transition relation*, $\eta : L \rightarrow \mathbb{Z}$ is the price function, T is the set of target locations, $T \cap L = \emptyset$; and $f_{goal} : T \rightarrow \mathcal{F}$ assigns external cost functions to target locations.

We refer to Player 1 as the controller and Player 2 as the environment. A priced timed game begins with a token placed on some initial location ℓ with valuation $\mathbf{0}$ and cost accumulated being so far being 0. At each round, the player who controls the current location ℓ chooses a delay t (to be elapsed in ℓ) and an outgoing transition $e = (\ell, g, r, \ell') \in X$ to be taken after t delay at ℓ . The clock valuation is then updated according to the delay t , the reset r , the cost is incremented by $\eta(\ell) \cdot t$ and the token is moved to the location ℓ' . The two players continue moving the token in this fashion, and give rise to a sequence of locations and transitions called a *play* of the game. A configuration or state of a PTG is a tuple (ℓ, ν, c) where $\ell \in L$ is a location, $\nu \in \mathbb{R}^{|\mathcal{C}|}$ is a valuation, and c is the cost accumulated from the start of the play. We assume, w.l.o.g [2], that the clock valuations are bounded.

► **Definition 2 (PTG semantics).** The semantics of a PTG \mathcal{G} is a labelled state-transition game arena $\llbracket \mathcal{G} \rrbracket = (\mathcal{S} = S_1 \uplus S_2, S_{init}, A, E, \pi, \kappa)$ where

- $S_j = L_j \times \mathbb{R}^{|\mathcal{C}|}$ are the Player j states with $\mathcal{S} = S_1 \uplus S_2$,
- $S_{init} \subseteq \mathcal{S}$ are initial states s.t. $(\ell, \nu) \in S_{init}$ if $\ell \in L_{init}$, $\nu = \mathbf{0}$,
- $A = \mathbb{R}_{\geq 0} \times X$ is the set of *timed moves*,
- $E : (\mathcal{S} \times A) \rightarrow \mathcal{S}$ is the transition function s.t. for $s = (\ell, \nu), s' = (\ell', \nu') \in \mathcal{S}$ and $\tau = (t, e) \in A$ the function $E(s, \tau)$ is defined if $e = (\ell, g, r, \ell')$ is a transition of the PTG and $\nu \in \llbracket g \rrbracket$; moreover $E(s, \tau) = s'$ if $\nu' = (\nu + t)[r:=0]$ (we write $s \xrightarrow{\tau} s'$ when $E(s, \tau) = s'$);
- $\pi : \mathcal{S} \times A \rightarrow \mathbb{R}$ is the price function such that $\pi((\ell, \nu), (t, e)) = \eta(\ell) \cdot t$; and
- $\kappa : \mathcal{S} \rightarrow \mathbb{R}$ is an external cost function such that $\kappa(\ell, \nu)$ is defined when $\ell \in T$ such that $\kappa(\ell, \nu) = f_{goal}(\ell)(\nu)$.

A *play* $\rho = \langle s_0, \tau_1, s_1, \tau_2, \dots, s_n \rangle$ is a finite sequence of states and actions s.t. $s_0 \in S_{init}$ and $s_i \xrightarrow{\tau_{i+1}} s_{i+1}$ for all $0 \leq i < n$. The infinite plays are defined in an analogous manner. For a finite play ρ we write its last state as $\text{last}(\rho) = s_n$. For a (infinite or finite) play ρ we write $\text{stop}(\rho)$ for the index of first target state and if it doesn't visit a target state then $\text{stop}(\rho) = \infty$. We denote the set of plays as $\text{Plays}_{\mathcal{G}}$. For a play $\rho = \langle s_0, (t_1, a_1), s_1, (t_2, a_2), \dots \rangle$ if $\text{stop}(\rho) = n < \infty$ then $\text{Cost}_{\mathcal{G}}(\rho) = \kappa(s_n) + \sum_{i=1}^n \pi(s_{i-1}, (t_i, a_i))$ else $\text{Cost}_{\mathcal{G}}(\rho) = +\infty$.

A *strategy* of player j in \mathcal{G} is a function $\sigma : \text{Plays}_{\mathcal{G}} \rightarrow A$ such that for a play ρ the function $\sigma(\rho)$ is defined if $\text{last}(\rho) \in S_j$. We say that a strategy σ is memoryless if $\sigma(\rho) = \sigma(\rho')$ when $\text{last}(\rho) = \text{last}(\rho')$, otherwise we call it memoryful. We write Strat_1 and Strat_2 for the set of strategies of player 1 and 2, respectively.

A play ρ is said to be *compatible to a strategy* σ of player $j \in \{1, 2\}$ if for every state s_i in ρ that belongs to Player j , $s_{i+1} = \sigma(s_i)$. Given a pair of strategies $(\sigma_1, \sigma_2) \in \text{Strat}_1 \times \text{Strat}_2$,

and a state s , the outcome of (σ_1, σ_2) from s denoted $\text{Outcome}(s, \sigma_1, \sigma_2)$ is the unique play that starts at s and is compatible with both strategies. Given a player 1 strategy $\sigma_1 \in \text{Strat}_1$ we define its cost $\text{Cost}_{\mathcal{G}}(s, \sigma_1)$ as $\sup_{\sigma_2 \in \text{Strat}_2} (\text{Cost}(\text{Outcome}(s, \sigma_1, \sigma_2)))$. We now define the *optimal reachability-cost* for Player 1 from a state s as

$$\text{OptCost}_{\mathcal{G}}(s) = \inf_{\sigma_1 \in \text{Strat}_1} \sup_{\sigma_2 \in \text{Strat}_2} (\text{Cost}(\text{Outcome}(s, \sigma_1, \sigma_2))).$$

A strategy $\sigma_1 \in \text{Strat}_1$ is said to be optimal from s if $\text{Cost}_{\mathcal{G}}(s, \sigma_1) = \text{OptCost}_{\mathcal{G}}(s)$. Since the optimal strategies may not always exist [5] we define ϵ optimal strategies. For $\epsilon > 0$ a strategy $\sigma_{\epsilon} \in \text{Strat}_1$ is called ϵ -optimal if $\text{OptCost}_{\mathcal{G}}(s) \leq \text{Cost}_{\mathcal{G}}(s, \sigma_{\epsilon}) < \text{OptCost}_{\mathcal{G}}(s) + \epsilon$. Given a PTG \mathcal{G} and a bound $K \in \mathbb{Z}$, the *cost-optimal reachability problem* for PTGs is to decide whether there exists a strategy for player 1 such that $\text{OptCost}_{\mathcal{G}}(s) \leq K$ from some starting state s .

► **Theorem 3** ([3]). *Cost-optimal reachability problem is undecidable for PTGs with 3 clocks.*

► **Theorem 4** ([5, 10, 12]). *The ϵ -optimal strategy is computable for 1 clock PTGs.*

3 Robust Semantics

Under the robust semantics of priced timed games the environment player—also called as the perturbator—is more privileged as it has the power to perturb any delay chosen by the controller by an amount in $[-\delta, \delta]$, where $\delta > 0$ is a pre-defined bounded quantity. However, in order to ensure time-divergence there is a restriction that the time delay at all locations of the RPTG must be $\geq \delta$. There are the following two perturbation semantics as defined in [7].

- *Excess semantics.* At any controller location, the time delay t chosen by the controller is altered to some $t' \in [t - \delta, t + \delta]$ by the perturbator. However, the constraints on the outgoing transitions of the controller locations are evaluated with respect to the time elapse t chosen by the controller. If the constraint is satisfied with respect to t , then the values of all variables which are not reset on the transition are updated with respect to t' ; the variables which are reset obtain value 0.
- *Conservative semantics.* In this, the constraints on the outgoing transitions are evaluated with respect to t' .

In both semantics, the delays chosen by perturbator at his locations are not altered, and the constraints on outgoing transitions are evaluated in the usual way, as in PTG.

A Robust-Priced Timed Automata (RPTA) is an RPTG which has only controller locations. At all these locations, for any time delay t chosen by controller, perturbator can implicitly perturb t by a quantity in $[-\delta, \delta]$. The excess as well as the conservative perturbation semantics for RPTA are defined in the same way as in the RPTG. Note that our RPTA coincides with that of [7] when the cost functions at all target locations are of the form $cf : \mathbb{R}_{\geq 0}^n \rightarrow \{0\}$. Our RPTG are turn-based, and have cost functions at the targets, while RPTGs studied in [7] are concurrent.

► **Definition 5** (Excess Perturbation Semantics). Let $\mathcal{R} = (L_1, L_2, L_{init}, \mathcal{C}, X, \eta, T, f_{goal})$ be a RPTG. Given a $\delta > 0$, the excess perturbation semantics of RPTG \mathcal{R} is a LTS $\llbracket \mathcal{R} \rrbracket = (\mathcal{S}, \mathcal{A}, \mathcal{E})$ where $\mathcal{S} = S_1 \cup S_2 \cup (T \times \mathbb{R}_{\geq 0})$, $\mathcal{A} = A_1 \cup A_2$ and $\mathcal{E} = E_1 \cup E_2$. We define the set of states, actions and transitions for each player below.

- $S_1 = L_1 \times \mathbb{R}^{|\mathcal{C}|}$ are the controller states,
- $S_2 = (L_2 \times \mathbb{R}^{|\mathcal{C}|}) \cup (S_1 \times \mathbb{R}_{\geq 0} \times X)$ are the perturbator states. The first kind of states are encountered at perturbator locations. The second kind of states are encountered when controller chooses a delay $t \in \mathbb{R}_{\geq 0}$ and a transition $e \in X$ at a controller location.
- $A_1 = \mathbb{R}_{\geq 0} \times X$ are controller actions
- $A_2 = (\mathbb{R}_{\geq 0} \times X) \cup [-\delta, \delta]$ are perturbator actions. The first kind of actions $(\mathbb{R}_{\geq 0} \times X)$ are chosen at states of the form $L_2 \times \mathbb{R}^{|\mathcal{C}|} \in S_2$, while the second kind of actions are chosen at states of the form $S_1 \times \mathbb{R}_{\geq 0} \times X \in S_2$,

- $E_1 = (S_1 \times A_1 \times S_2)$ is the set of controller transitions such that for a controller state (l, ν) and a controller action (t, e) , $E_1((l, \nu), (t, e))$ is defined iff there is a transition $e = (l, g, a, r, l')$ in \mathcal{R} such that $\nu + t \in \llbracket g \rrbracket$.
- $E_2 = S_2 \times A_2 \times (S_1 \cup S_2 \cup (T \times \mathbb{R}_{\geq 0}))$ is the set of perturbator transitions such that
 - For a perturbator state of the type (l, ν) and a perturbator action (t, e) , we have $(l', \nu') = E_2((l, \nu), (t, e))$ iff there is a transition $e = (l, g, a, r, l')$ in \mathcal{R} such that $\nu + t \in \llbracket g \rrbracket$, $\nu' = (\nu + t)[r := 0]$,
 - For a perturbator state of type $((l, \nu), t, e)$ and a perturbator action $\varepsilon \in [-\delta, \delta]$, we have $(l', \nu') = E_2(((l, \nu), t, e), \varepsilon)$ iff $e = (l, g, a, r, l')$, and $\nu' = (\nu + t + \varepsilon)[r := 0]$.

We now define the cost of the transitions, denoted as $\text{Cost}(t, e)$ as follows :

- For controller transitions : $(l, \nu) \xrightarrow{(t, e)} ((l, \nu), t, e)$: the cost accumulated is $\text{Cost}(t, e) = 0$.
- For perturbator transitions :
 - From perturbator states of type (l, ν) : $(l, \nu) \xrightarrow{t, e} (l', \nu')$, the cost accumulated is $\text{Cost}(t, e) = t * \eta(l)$.
 - From perturbator states of type $((l, \nu), t, e)$: $((l, \nu), t, e) \xrightarrow{\varepsilon} (l', \nu')$, the cost accumulated is $(t + \varepsilon) * \eta(l)$. Note that although this transition has no edge choice involved and the perturbation delay chosen is $\varepsilon \in [-\delta, \delta]$, the controller action (t, e) chosen in the state (l, ν) comes into effect in this transition. Hence for the sake of uniformity, we denote the cost accumulated in this transition to be $\text{Cost}(t + \varepsilon, e) = (t + \varepsilon) * \eta(l)$.

Note that we check satisfiability of the constraint g before the perturbation; however, the reset occurs after the perturbation. The notions of a path and a winning play are the same as in PTG. We shall now adapt the definitions of cost of a play, and a strategy for the excess perturbation semantics. Let $\rho = \langle s_1, (t_1, e_1), s_2, (t_2, e_2), \dots, (t_{n-1}, e_{n-1}), s_n \rangle$ be a path in the LTS $\llbracket \mathcal{R} \rrbracket$. Given a $\delta > 0$, for a finite play ρ ending in target location, we define $\text{Cost}_{\mathcal{R}}^{\delta}(\rho) = \sum_{i=1}^n \text{Cost}(t_i, e_i) + f_{\text{goal}}(l_n)(\nu_n)$ as the sum of the costs of all transitions as defined above along with the value from the cost function of the target location l_n . Also, we re-define the cost of a strategy σ_1 from a state s for a given $\delta > 0$ as $\text{Cost}_{\mathcal{R}}^{\delta}(s, \sigma_1) = \sup_{\sigma_2 \in \text{Strat}_2(\mathcal{R})} \text{Cost}_{\mathcal{R}}^{\delta}(\text{Outcome}(s, \sigma_1, \sigma_2))$. Similarly, $\text{OptCost}_{\mathcal{R}}^{\delta}$ is the optimal cost under excess perturbation semantics for a given $\delta > 0$ defined as

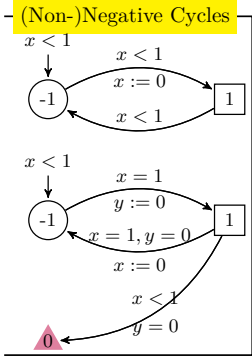
$$\text{OptCost}_{\mathcal{R}}^{\delta}(s) = \inf_{\sigma_1 \in \text{Strat}_1(\mathcal{R})} \sup_{\sigma_2 \in \text{Strat}_2(\mathcal{R})} (\text{Cost}_{\mathcal{R}}^{\delta}(\text{Outcome}(s, \sigma_1, \sigma_2))).$$

Since optimal strategies may not always exist, we define ϵ -optimal strategies such that for every $\epsilon > 0$, $\text{OptCost}_{\mathcal{R}}^{\delta}(s) \leq \text{Cost}_{\mathcal{R}}^{\delta}(s, \sigma_1) < \text{OptCost}_{\mathcal{R}}^{\delta}(s) + \epsilon$. Given a δ and a RPTG \mathcal{R} with a single clock x , a strategy σ_1 is called (ϵ, N) -acceptable [5] for $\epsilon > 0, N \in \mathbb{N}$ when (1) it is memoryless, (2) it is ϵ -optimal and (3) there exist N consecutive intervals $(I_i)_{1 \leq i \leq N}$ partitioning $[0, 1]$ such that for every location l , for every $1 \leq i \leq N$ and every integer $\alpha < M$ (where M is the maximum bound on the clock value), the function that maps the clock values $\nu(x)$ to the cost of the strategy σ_1 at every state $(l, \nu(x))$, $(\nu(x) \mapsto \text{Cost}_{\mathcal{R}}^{\delta}((l, \nu(x)), \sigma_1))$ is affine for every interval $\alpha + I_i$. Also, the strategy σ_1 is constant over the values $\alpha + I_i$ at all locations, that is, when $\nu(x) \in \alpha + I_i$, the strategy $\sigma_1(l, \nu(x))$ is constant. The number N is an important attribute of the strategy as it establishes that the strategy does not fluctuate infinitely often and is implementable.

Now, we shall define limit variations of costs, strategies and values as $\delta \rightarrow 0$. The *limit-cost* of a controller strategy σ_1 from state s is defined over all plays ρ starting from s that are compatible with σ_1 as:

$$\text{LimCost}_{\mathcal{R}}(s, \sigma_1) = \lim_{\delta \rightarrow 0} \sup_{\sigma_2 \in \text{Strat}_2(\mathcal{R})} \text{Cost}_{\mathcal{R}}^{\delta}(\text{Outcome}(s, \sigma_1, \sigma_2)).$$

The *limit strategy upper-bound problem* [7] for excess perturbation semantics asks, given a RPTG \mathcal{R} , state $s = (l, \mathbf{0})$ with cost 0 and a rational number K , whether there exists a strategy σ_1 such that $\text{LimCost}_{\mathcal{R}}(s, \sigma_1) \leq K$. The following are the main results of [7].



- **Theorem 6** (Known results [7]). 1. The limit-strategy upper-bound problem is undecidable for RPTA and RPTG under excess perturbation semantics, for ≥ 10 clocks.
2. For a fixed $\delta \in [0, \frac{1}{3}]$, and a given RPTA \mathcal{A} , a target location l and a rational K , it is undecidable whether $\inf_{\sigma_1} \sup_{\sigma_2} \text{cost}_{\sigma_1, \sigma_2}(\rho) < K$ such that ρ ends in l . $\text{cost}_{\sigma_1, \sigma_2}(\rho)$ is the cost of the unique run ρ obtained from the pair of strategies (σ_1, σ_2) .

We consider a semantic subclass of RPTGs in which the accumulated cost of any cycle is non-negative: that is, any iteration of a cycle will always have a non-negative cost. Consider the two cycles depicted. The one on top has a non-negative cost, while the one below always has a negative cost. In the cycle below, the perturbator will not perturb, since that will lead to a target state. In the rest of the paper, we consider this semantic class of RPTGs (RPTAs), and prove decidability and undecidability results; however, we will refer to them as RPTGs(RPTAs). Our key contributions are the following theorems.

► **Theorem 7.** The limit-strategy upper-bound problem is undecidable for RPTA with 5 clocks, location prices in $\{0, 1\}$, and cost functions $\text{cf} : \mathbb{R}_{\geq 0}^n \rightarrow \{0\}$ at all target locations.

► **Theorem 8.** Given a 1-clock RPTG \mathcal{R} and a $\delta > 0$, we can compute $\text{OptCost}_{\mathcal{R}}^{\delta}(s)$ for every state $s = (l, \nu)$. For every $\epsilon > 0$, there exists an $N \in \mathbb{N}$ such that the controller has an (ϵ, N) -acceptable strategy.

The rest of the paper is devoted to the proof sketches of these two theorems, while we give detailed proofs in the appendix.

4 Undecidability with 5 clocks

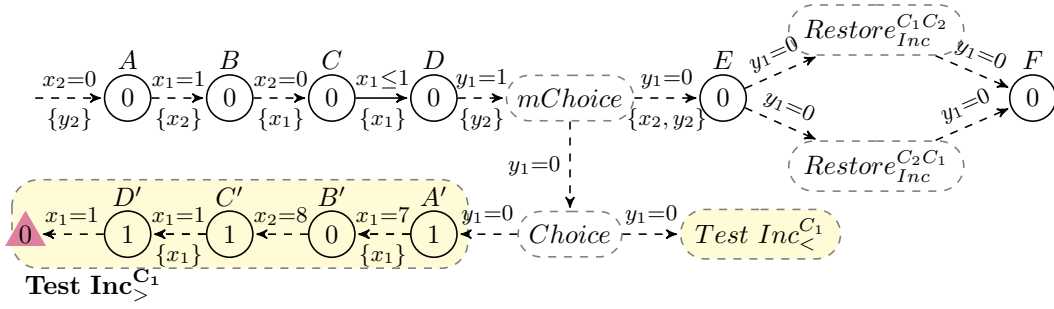
In this section, we improve the result of [7] by showing that the limit strategy upper bound problem is undecidable for robust priced timed automata with 5 or more clocks. The undecidability result is obtained using a reduction to the halting problem of two-counter machines.

A two-counter machine has counters C_1 and C_2 , and a list of instructions I_1, I_2, \dots, I_n , where I_n is the *halt instruction*. For each $1 \leq i \leq n-1$, I_i is one of the following instructions: **increment** c_b : $c_b := c_b + 1$; **goto** I_j , for $b = 1$ or 2 , **decrement** c_b **with zero test**: **if** $(c_b = 0)$ **goto** I_j **else** $c_b := c_b - 1$; **goto** I_j , where c_1, c_2 represent the counter values. The initial values of both counters are 0. Given the initial configuration $(I_1, 0, 0)$ the halting problem for two counter machines is to find if the configuration (I_n, c_1, c_2) is reachable, with $c_1, c_2 \geq 0$. This problem is known to be undecidable.

We simulate the two counter machine using a RPTA with 5 clocks x_1, z, x_2, y_1 and y_2 under the excess perturbation semantics. The counters are encoded in clocks x_1 and z as $x_1 = \frac{1}{2^i} + \varepsilon_1$ and $z = \frac{1}{2^j} + \varepsilon_2$ where i, j are respectively the values of counters C_1, C_2 , and ε_1 and ε_2 denote accumulated values due to possible perturbations. Clocks x_2, y_1 and y_2 help with the rough work. The simulation is achieved as follows: for each instruction, we have a module simulating it. Upon entering the module, the clocks are in their normal form i.e. $x_1 = \frac{1}{2^i} + \varepsilon_1, z = \frac{1}{2^j} + \varepsilon_2$ and $x_2 = 0$ and $y_1 = y_2 = 0$.

4.1 Increment module

The module in Figure 1 simulates the increment of counter C_1 . The value of counter C_2 remains unchanged since the value of clock z remains unchanged at the exit from the module. Upon entering A the clock values are $x_1 = \frac{1}{2^i} + \varepsilon_1, z = \frac{1}{2^j} + \varepsilon_2, x_2 = y_1 = y_2 = 0$. Here ε_1 and ε_2 respectively denote the perturbations accumulated so far. We denote by α , the value of clock x_1 , i.e. $\frac{1}{2^i} + \varepsilon_1$. Thus at A , the delay is $1 - \alpha$. Note that the dashed edges are unperturbed (this is a short hand notation. A small gadget that implements this is described in Appendix B), so $x_1 = 1$ on entering B . No time elapse happens at B , and at C , controller



■ **Figure 1 Increment C_1 module** : The module keeps the fractional part of the clock z unchanged. The dashed edges represent unperturbed edges (detailed in Appendix B).

chooses a delay t . This t must be $\frac{\alpha}{2}$ to simulate the increment correctly. t can be perturbed by an amount δ by the perurbator, where δ can be both positive or negative, obtaining $x_2 = t + \delta, x_1 = 0, y_1 = 1 - \alpha + t + \delta$ on entering D . At D , the delay is $\alpha - t - \delta$. Thus the total delay from the entry point A in this module to the $mChoice$ module is 1 time unit. At the entry of the $mChoice$ ($mChoice$ and $Restore$ modules are in Appendix B) module, the clock values are $x_1 = \alpha - t - \delta, z = 1 + \frac{1}{2^j} + \varepsilon_2, x_2 = \alpha, y_1 = 1, y_2 = 0$. To correctly simulate the increment of C_1 , t should be exactly $\frac{\alpha}{2}$.

At the $mChoice$ module, perturbator can either continue the simulation (by going through the $Restore$ module) or verify the correctness of controller's delay (check $t = \frac{\alpha}{2}$). The $mChoice$ module adds 3 units to the values of x_1, x_2 and z , and resets y_1, y_2 . Due to the $mChoice$ module, the clock values are $x_1 = 3 + \alpha - t - \delta, z = 4 + \frac{1}{2^j} + \varepsilon_2, x_2 = 3 + \alpha, y_1 = 1, y_2 = 0$. If perturbator chooses to continue the simulation, then $Restore$ module brings all the clocks back to normal form. Hence upon entering F , the clock values are $x_1 = \alpha - t - \delta, z = \frac{1}{2^j} + \varepsilon_2, x_2 = y_1 = 1, y_2 = 0$. This value of x_1 is $\frac{\alpha}{2} + \varepsilon_1$, since $t = \frac{\alpha}{2}$ and $\varepsilon_1 = -\delta$, the perturbation effect.

Let us now see how perturbator verifies $t = \frac{\alpha}{2}$ by entering the $Choice$ module. The $Choice$ module also adds 3 units to the values of x_1, x_2 and z , and resets y_1, y_2 . The module $Test Inc_{>}^{C_1}$ is invoked to check if $t > \frac{\alpha}{2}$, and the module $Test Inc_{<}^{C_1}$ is invoked to check if $t < \frac{\alpha}{2}$. Note that using the $mChoice$ module and the $Choice$ module one after the other, the clock values upon entering $Test Inc_{>}^{C_1}$ or $Test Inc_{<}^{C_1}$ are $x_1 = 6 + \alpha - t - \delta, z = 7 + \frac{1}{2^j} + \varepsilon_2, x_2 = 6 + \alpha, y_1 = 0, y_2 = 0$.

Test Inc $_{>}^{C_1}$: The delay at A' is $1 - \alpha + t + \delta$, obtaining $x_2 = 7 + t + \delta$, and the cost accumulated is $1 - \alpha + t + \delta$. At B' , $1 - t - \delta$ time is spent, obtaining $x_1 = 1 - t - \delta$. Finally, at C' , a time $t + \delta$ is spent, and at D' , one time unit, making the total cost accumulated $2 - \alpha + 2t + 2\delta$ at the target location. The cost function at the target assigns the cost 0 for all valuations, hence the total cost to reach the target is $2 + 2t - \alpha + 2\delta$ which is greater than $2 + 2\delta$ iff $2t - \alpha > 0$, i.e. iff $t > \frac{\alpha}{2}$.

► **Lemma 9.** Assume that an increment C_b ($b \in \{0, 1\}$) module is entered with the clock valuations in their normal forms. Then controller has a strategy to reach either location l_j corresponding to instruction I_j of the two-counter machine or a target location is reached with cost at most $2 + |2\delta|$, where δ is the perturbation added by perturbator.

4.2 Complete Reduction

The entire reduction consists of constructing a module corresponding to each instruction I_i , $1 \leq i \leq n$, of the two-counter machine. The first location of the module corresponding to instruction I_1 is the initial location. We simulate the halting instruction I_n by a target location

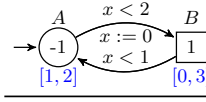
with cost function $cf : \mathbb{R}_{\geq 0}^5 \rightarrow \{0\}$. We denote the robust timed automaton simulating the two counter machine by $\tilde{\mathcal{A}}$, s is the initial state $(l, \mathbf{0}, \mathbf{0})$.

► **Lemma 10.** *The two counter machine halts if and only if there is a strategy σ of controller such that $\text{limcost}_{\tilde{\mathcal{A}}}(\sigma, s) \leq 2$.*

The details of the decrement and zero test modules are in Appendix B. They are similar to the increment module; if player 2 desires to verify the correctness of player 1's simulation, a cost $> 2 + |2\delta|$ is accumulated on reaching a target location iff player 1 cheats. In the limit, as $\delta \rightarrow 0$, the limcost will be > 2 iff controller cheats. The other possibility to obtain a limcost > 2 is when the two counter machine does not halt.

5 Decidability of One-clock RPTG

A Dwell-time PTG

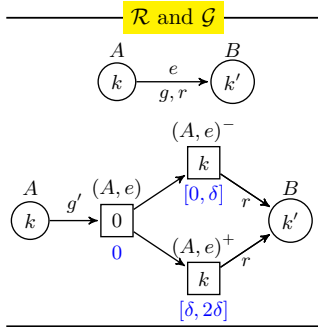


In order to show the decidability of the optimal reachability game for 1 clock RPTG \mathcal{R} and a fixed $\delta > 0$, we perform a series of reachability and optimal cost preserving transformations. The idea is to reduce the RPTG into a simpler priced timed game, while preserving the optimal costs. The advantages of this conversion is that the semantics of PTGs are easier to understand, and one could adapt known algorithms to solve PTGs.

On the other hand, the PTGs that we obtain are 1-clock PTGs with dwell-time requirement (having restrictions on minimum as well as maximum amount of time spent at certain locations), see for example, a dwell-time PTG with two locations A, B . A minimum of 1 and a maximum of two units of time should be spent at A , while a maximum of 3 time units can be spent at B . If we wish to model this using standard PTGs, we need one extra clock and we can not use the decidability results of 1 clock PTG to show the decidability of our model. We show in Section 5.4 how to solve 1-clock PTGs with dwell-time requirements.

Our transformations are as follows: (i) for a given δ , our first transformation reduces the RPTG \mathcal{R} into a dwell-time PTG \mathcal{G} (Section 5.1); (ii) our second transformation restricts to dwell-time PTGs where the clock is bounded by $1 + \delta$. To achieve this, we use a notion of *fractional resets*, and denote these PTGs as $\mathcal{G}_{\mathcal{F}}$ (Section 5.2); (iii) our third and last transformation restricts $\mathcal{G}_{\mathcal{F}}$ without resets (Section 5.3). The reset-free dwell-time PTG is denoted $\mathcal{G}_{\mathcal{F}}'$. For each transformation, we prove that the optimal cost in each state of the original game is the same as the optimal cost at some corresponding state of the new game. We also show that an (ϵ, N) -strategy of the original game can be computed from some (ϵ', N') -strategy in the new game. The details of each transformation and correctness is established in subsequent sections. We then solve $\mathcal{G}_{\mathcal{F}}'$ employing a technique inspired by [5] while ensuring that the robust semantics are satisfied.

5.1 Transformation 1: RPTG \mathcal{R} to dwell-time PTG \mathcal{G}



Given a one clock RPTG $\mathcal{R} = (L_1, L_2, \{x\}, X, \eta, T, f_{goal})$ and a $\delta > 0$, we construct a dwell-time PTG $\mathcal{G} = (L_1, L_2 \cup L', \{x\}, X', \eta', T, f_{goal})$. All the controller, perturbator locations of \mathcal{R} (L_1 and L_2) are carried over respectively as player 1, player 2 locations in \mathcal{G} . In addition, we have some new player 2 locations L' in \mathcal{G} . The dwell-time PTG \mathcal{G} constructed has dwell-time restrictions for the new player 2 locations L' . The locations of L' are either urgent, or have a dwell-time of $[\delta, 2\delta]$ or $[0, \delta]$. All the perturbator transitions of \mathcal{R} are retained as it is in \mathcal{G} . Every transition in \mathcal{R} from a controller location A to some location B is replaced in \mathcal{G} by a game graph as shown. Let $e = (A, g, r, B)$ be the transition from a controller location A to a location B with guard g , and reset r . Depending on the guard g , in the transformed game graph, we have the new guard g' . If g is $x = H$, then g' is $x = H - \delta$, while if g is $H < x < H + 1$, then g' is $H - \delta < x < H + 1 - \delta$, for $H > 0$. When g is $0 < x < K$, then g' is $0 \leq x < K - \delta$ and $x = 0$

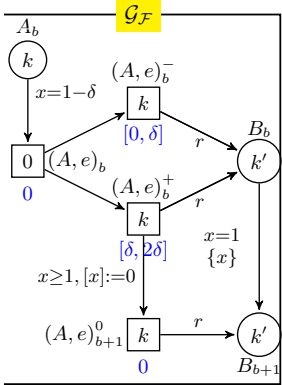
stays unchanged. It can be seen that doing this transformation to all the controller edges of a RPTG \mathcal{R} gives rise to a dwell-time PTG \mathcal{G} .

Lets consider the transition from A to B in \mathcal{R} . Assume that the transition from A to B (called edge e) had a constraint $x = 1$, and assume that $x = \nu$ on entering A . Then, in \mathcal{R} , controller elapses a time $1 - \nu$, and reaches B ; however on reaching B , the value of x is in the range $[1 - \delta, 1 + \delta]$ depending on the perturbation. Also, the cost accumulated at A is $k * (1 - \nu + \gamma)$, where $\gamma \in [-\delta, \delta]$. To take into consideration these semantic restrictions of \mathcal{R} , we transform the RPTG \mathcal{R} into a dwell-time PTG \mathcal{G} . First of all, we change the constraint $x = 1$ into $x = 1 - \delta$ from A (a player 1 location) and enter a new player 2 location (A, e) . This player 2 location is an urgent location. The correct strategy for player 1 is to spend a time $1 - \nu - \delta$ at A (corresponding to the time $1 - \nu$ he spent at A in \mathcal{R}). At (A, e) , player 2 can either proceed to one of the player 2 locations $(A, e)^-$ or $(A, e)^+$. The player 2 location (A, e) models perturbator's choices of positive or negative perturbation in \mathcal{R} . If player 2 goes to $(A, e)^-$, then on reaching B , the value of x is in the interval $[1 - \delta, 1]$ (this corresponds to perturbator's choice of $[-\delta, 0]$ in \mathcal{R}) and if he goes to $(A, e)^+$, then the value of x at B is in the interval $[1, 1 + \delta]$ (this corresponds to perturbator's choice of $[0, \delta]$ in \mathcal{R}). The reset happening in the transition from A to B in \mathcal{R} is now done on the transition from $(A, e)^-$ to B and from $(A, e)^+$ to B . Thus, note that the possible ranges of x as well as the accumulated cost in \mathcal{R} while reaching B are preserved in the transformed dwell-time PTG.

► **Lemma 11.** *Let \mathcal{R} be a RPTG and \mathcal{G} be the corresponding dwell-time PTG obtained using the transformation above. Then for every state s in \mathcal{R} , $\text{OptCost}_{\mathcal{R}}(s) = \text{OptCost}_{\mathcal{G}}(s)$. An (ϵ, N) -strategy in \mathcal{R} can be computed from a (ϵ, N) -strategy in \mathcal{G} and vice versa.*

Proof In Appendix C.

5.2 Transformation 2: Dwell-time PTG \mathcal{G} to Dwell-time FRPTG $\mathcal{G}_{\mathcal{F}}$



Recall that the locations of the dwell-time PTG \mathcal{G} is $L_1 \cup L_2 \cup L'$ where $L_1 \cup L_2$ are the set of locations of \mathcal{R} , and L' are new player 2 locations introduced in \mathcal{G} . In this section, we transform the dwell-time PTG \mathcal{G} into a dwell-time PTG $\mathcal{G}_{\mathcal{F}}$ having the restriction that the value of x is in $[0, 1]$ at all locations corresponding to $L_1 \cup L_2$, and is in $[0, 1 + \delta]$ at all locations corresponding to L' . While this transformation is the same as that used in [5], the main difference is that we introduce special resets called *fractional resets* which reset only the integral part of clock x while its fractional part is retained. For instance, if the value of x was 1.3, then the operation $[x] := 0$ makes the value of x to be 0.3.

Given a one clock, dwell-time PTG $\mathcal{G} = (L_1, L_2 \cup L', \{x\}, X, \eta, T, f_{goals})$ with M being the maximum value that can be assumed by clock x , we define a dwell-time PTG with fractional resets (FRPTG) $\mathcal{G}_{\mathcal{F}}$. In $\mathcal{G}_{\mathcal{F}}$, we have $M + 1$ copies of the locations in $L_1 \cup L_2$ as well as the locations in L' with dwell time $[0, \delta]$, $[0, 0]$.

These $M + 1$ copies of L' have the same dwell-time restrictions in $\mathcal{G}_{\mathcal{F}}$. The copies are indexed by $i, 0 \leq i \leq M$, capturing the integral part of clock x in \mathcal{G} . Finally, we have in \mathcal{G} , the locations of L' with dwell-time restriction $[\delta, 2\delta]$. For each such location $(A, e)^+$, we have in $\mathcal{G}_{\mathcal{F}}$, the locations $(A, e)_i^+$ and $(A, e)_{i+1}^0$ for $0 \leq i \leq M$. The dwell-time restriction for $(A, e)_i^+$ is same as $(A, e)^+$, while locations $(A, e)_{i+1}^0$ are urgent. The prices of locations are carried over as they are in the various copies.

The transitions in $\mathcal{G}_{\mathcal{F}}$ consists of the following: (1) $l_i \xrightarrow{(g-i) \cap 0 \leq x < 1} m_i^1$ if $l \xrightarrow{g} m \in X$; (2) $l_i \xrightarrow{(g-i) \cap 0 \leq x < 1; \{x\}} m_0$ if $l \xrightarrow{g; \{x\}} m \in X$; (3) $l_i \xrightarrow{x=1, \{x\}} l_{i+1}$, for $l \in L_1 \cup L_2$, and $(A, e)_i^+ \xrightarrow{x \geq 1, [x] := 0} (A, e)_{i+1}^0$ for $i < M$. Consider for example, the constraint g' between

¹ $g - i$ represents the constraint obtained by shifting the constraint by $-i$

A and (A, e) as $x = (b + 1) - \delta$ in \mathcal{G} . Then the value of x is $b + (1 - \delta)$ for $b < M$ when $(A, e)^+$ is entered in \mathcal{G} . The location $(A, e)^+$ with $\nu(x) = b + (1 - \delta)$ is represented in $\mathcal{G}_{\mathcal{F}}$ as $(A, e)_b^+$ with $\nu(x) = 1 - \delta$. If player 2 spends $[\delta, 2\delta]$ time at $(A, e)^+$ in \mathcal{G} , then $\nu(x) \in [b + 1, b + 1 + \delta]$. If there are no resets to goto B , then $\nu(x) \in [b + 1, (b + 1) + \delta]$ at B . Correspondingly in $\mathcal{G}_{\mathcal{F}}$, $\nu(x) \in [1, 1 + \delta]$ at $(A, e)_b^+$. By construction, B_b is not reachable, since we check $0 \leq x < 1$ on the transition to B_b . The fractional reset is employed to obtain $x = \delta$ while moving to $(A, e)_{b+1}^0$. This ensures that $x = \delta$ on reaching B_{b+1} , thereby preserving the perturbation, and keeping $x < 1$. A normal reset would have destroyed the value obtained by perturbation. The mapping f between states of \mathcal{G} and $\mathcal{G}_{\mathcal{F}}$ is as follows: $f(l, x) = (l_b, x - b)$, $b < M$, and $x \in [b, b + 1]$, $l \in L_1 \cup L_2$, $f((A, e), x) = ((A, e)_b, x - b)$, $b < M$, and $x \in [b, b + 1]$, $f((A, e)^-, x) = ((A, e)_b^-, x - b)$, $b < M$, and $x \in [b, b + 1]$. Finally, $f((A, e)^+, x) = ((A, e)_b^+, x - b)$, $b < M$, and $x \in [b, b + 1] \cup [b + 1, b + 2]$. Note that in the last case, the value of $x - b$ can exceed 1 but is less than or equal to $1 + \delta$.

► **Lemma 12.** *For every state (l, ν) in \mathcal{G} , $\text{OptCost}_{\mathcal{G}}(l, \nu)$ in \mathcal{G} is the same as $\text{OptCost}_{\mathcal{G}_{\mathcal{F}}}(f(l, \nu))$ in $\mathcal{G}_{\mathcal{F}}$. For every $\epsilon > 0$, $N \in \mathbb{N}$, an (ϵ, N) -acceptable strategy in \mathcal{G} can be computed from an (ϵ, N) -acceptable strategy in $\mathcal{G}_{\mathcal{F}}$ and vice versa.*

5.3 Transformation 3: Dwell-time FRPTG $\mathcal{G}_{\mathcal{F}}$ to resetfree FRPTG $\mathcal{G}_{\mathcal{F}}'$

We now apply the final transformation to the FRPTG $\mathcal{G}_{\mathcal{F}}$ and construct a reset-free version of the FRPTG denoted $\mathcal{G}_{\mathcal{F}}'$. Assume that there are a total of n resets (including fractional resets) in the FRPTG. $\mathcal{G}_{\mathcal{F}}'$ consists of $n+1$ copies of the FRPTG: $\mathcal{G}_{\mathcal{F}0}, \mathcal{G}_{\mathcal{F}1}, \dots, \mathcal{G}_{\mathcal{F}n}$. Given the locations L of the FRPTG, the locations of $\mathcal{G}_{\mathcal{F}i}$ are L^i , $0 \leq i \leq n$. $\mathcal{G}_{\mathcal{F}0}$ starts with l^0 , where l is the initial location of the FRPTG and continues until a resetting transition happens. At the first resetting transition, $\mathcal{G}_{\mathcal{F}0}$ makes a transition to $\mathcal{G}_{\mathcal{F}1}$. The n th copy is directed to a sink target location S with cost function $cf : \mathbb{R}_{\geq 0} \rightarrow \{+\infty\}$ on the $(n+1)$ th reset. Note that each $\mathcal{G}_{\mathcal{F}i}$ is reset-free. One crucial property of each $\mathcal{G}_{\mathcal{F}i}$ is that on entering with some value of x in $[0, \delta]$, the value of x only increases as the transitions go along in $\mathcal{G}_{\mathcal{F}i}$; moreover, $x \leq 1 + \delta$ in each $\mathcal{G}_{\mathcal{F}i}$ by construction. The formal details and proof of Lemma 13 can be found in Appendix E.

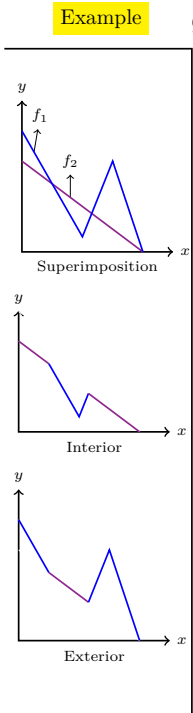
Using the cost function of S and those of the targets, we compute the optimal cost functions for all the locations of the deepest component $\mathcal{G}_{\mathcal{F}n}$. The cost functions of the locations of $\mathcal{G}_{\mathcal{F}i}$ are used to compute that of $\mathcal{G}_{\mathcal{F}i-1}$, and so on until the cost function of l^0 , the starting location of $\mathcal{G}_{\mathcal{F}0}$ is computed. An example can be seen in Appendix F.

► **Lemma 13.** *For every state (l, ν) in $\mathcal{G}_{\mathcal{F}}$, $\text{OptCost}_{\mathcal{G}_{\mathcal{F}}}(l, \nu) = \text{OptCost}_{\mathcal{G}_{\mathcal{F}}'}(l^0, \nu)$, where $\mathcal{G}_{\mathcal{F}}'$ is the resetfree FRPTG. For every $\epsilon > 0$, $N \in \mathbb{N}$, given an (ϵ, N) -acceptable strategy σ' in $\mathcal{G}_{\mathcal{F}}'$, we can compute a $(2\epsilon, N)$ -acceptable strategy σ in $\mathcal{G}_{\mathcal{F}}$ and vice versa.*

5.4 Solving the Resetfree FRPTG

Before we sketch the details, let us introduce some key notations. Observe that after our simplifying transformations, the cost functions cf are piecewise-affine continuous functions that assign a value to every valuation $x \in [0, 1 + \delta]$ (construction of FRPTG ensures $x \leq 1 + \delta$ always). The *interior* of two cost functions f_1 and f_2 is a cost function $f_3 : [0, 1 + \delta] \rightarrow \mathbb{R}$ defined by $f_3(x) = \min(f_1(x), f_2(x))$. Similarly, the *exterior* of f_1 and f_2 is a cost function $f_4 : [0, 1 + \delta] \rightarrow \mathbb{R}$ defined as $f_4(x) = \max(f_1(x), f_2(x))$. Clearly, f_3 and f_4 are also piecewise-affine continuous. The interior and exterior can be easily computed by *superimposing* f_1 and f_2 as shown graphically in the example by computing lower envelope and upper envelope respectively.

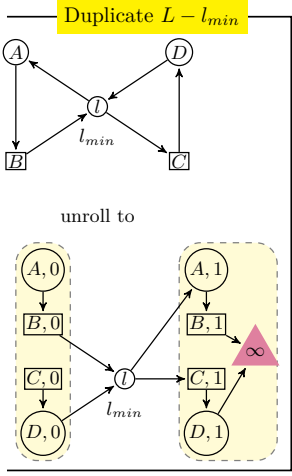
We now work on the reset-free components $\mathcal{G}_{\mathcal{F}i}$, and give an algorithm to compute $\text{OptCost}_{\mathcal{G}_{\mathcal{F}i}}(l, \nu)$ for every state (l, ν) of $\mathcal{G}_{\mathcal{F}i}$, $\nu(x) \in [0, 1 + \delta]$. We also show the existence of an N such that for any $\epsilon > 0$, and every $l \in L^i$, $\nu(x) \in [0, 1 + \delta]$, an (ϵ, N) -acceptable strategy can be computed. Consider the location of $\mathcal{G}_{\mathcal{F}i}$ that has the smallest price and



call it l_{min} . If this is a player 1 location, then intuitively, player 1 would want to spend as much time as possible here, and if this is a player 2 location, then player 2 would want to spend as less time as possible here. By our assumption, all the cycles in $\mathcal{G}_{\mathcal{F}_i}$ are non-negative, and hence if l_{min} is part of a cycle, revisiting it will only increase the total cost if at all. Player 1 thus would like to spend all the time he wants to during the first visit itself. We now prove that this is indeed the case. We consider two cases separately.

5.4.1 l_{min} is a Player 1 location

We split $\mathcal{G}_{\mathcal{F}_i}$ such that l_{min} is visited only once. We transform $\mathcal{G}_{\mathcal{F}_i}$ into $\mathcal{G}_{\mathcal{F}''}$ which has two copies of all locations except l_{min} such that corresponding to every location $l \neq l_{min}$, we have the copies $(l, 0)$ and $(l, 1)$. A special target location S is added with cost function assigning $+\infty$ to all clock valuations.



Given the transitions X of $\mathcal{G}_{\mathcal{F}_i}$, the FRPTG $\mathcal{G}_{\mathcal{F}''}$ has the following transitions.

- if $l \xrightarrow{g} l' \in X$ and $l, l' \neq l_{min}$ then $(l, 0) \xrightarrow{g} (l', 0)$ and $(l, 1) \xrightarrow{g} (l', 1)$
- if $l \xrightarrow{g} l' \in X$ and $l' = l_{min}$ then $(l, 0) \xrightarrow{g} l_{min}$ and $(l, 1) \xrightarrow{g} S$,
- if $l_{min} \xrightarrow{g} l$, then $l_{min} \xrightarrow{g} (l, 1)$

► **Lemma 14.** *For every state (l, ν) if $\nu \in [0, 1 + \delta]$ and $l \neq l_{min}$, we have that*

$$OptCost_{\mathcal{G}_{\mathcal{F}_i}}(l, \nu) = OptCost_{\mathcal{G}_{\mathcal{F}''}}((l, 0), \nu) \quad \text{and} \quad OptCost_{\mathcal{G}_{\mathcal{F}_i}}(l_{min}, \nu) = OptCost_{\mathcal{G}_{\mathcal{F}''}}(l_{min}, \nu).$$

We give an intuition for Lemma 14. Locations $(l, 0)$ have all the transitions available to location l in $\mathcal{G}_{\mathcal{F}_i}$. Also, any play in $\mathcal{G}_{\mathcal{F}''}$ which is compatible with a winning strategy of player 1 in $\mathcal{G}_{\mathcal{F}_i}$ contains only one of the locations $(l, 0), (l, 1)$ by construction of $\mathcal{G}_{\mathcal{F}''}$. The outcomes from $(l, 0)$ are more favourable than $(l, 1)$ for l as a player 1 location. Based on these intuitions, we conclude that $OptCost_{\mathcal{G}_{\mathcal{F}_i}}(l, \nu)$ is same as that for $((l, 0), \nu)$. This observation also leads to the ϵ -optimal strategy being the same as that for $(l, 0)$. Given a strategy σ' in $\mathcal{G}_{\mathcal{F}''}$, we construct σ in $\mathcal{G}_{\mathcal{F}_i}$ as $\sigma(l, \nu) = \sigma'((l, 0), \nu)$. Further, any strategy that revisits l_{min} in $\mathcal{G}_{\mathcal{F}_i}$ cannot be winning for player 1, since all cycles are non-negative; we end up at S with cost ∞ in $\mathcal{G}_{\mathcal{F}''}$. However, all strategies that do not revisit l_{min} in $\mathcal{G}_{\mathcal{F}_i}$ are preserved in $\mathcal{G}_{\mathcal{F}''}$, and hence $OptCost_{\mathcal{G}_{\mathcal{F}_i}}(l_{min}, \nu) = OptCost_{\mathcal{G}_{\mathcal{F}''}}(l_{min}, \nu)$. We iteratively solve the part of $\mathcal{G}_{\mathcal{F}''}$ with locations indexed 1 (i.e; $(l, 1)$) in the same fashion (picking minimal price locations) each time obtaining a smaller PTG. Computing the cost function of the minimal price location of the last such PTG, and propagating this backward, we compute the cost function of l_{min} . We then use the cost function of l_{min} to solve the part of $\mathcal{G}_{\mathcal{F}''}$ with locations indexed 0 (i.e; $(l, 0)$).

Computing the Optcost function of l_{min} : Algorithm 1 computes the optcost function for a player 1 location l_{min} , assuming all the constraints on outgoing transitions from l_{min} are the same, namely $x \in [0, 1]$. We discuss adapting the algorithm to work for transitions with different constraints in Appendix G. A few words on the notation used: if a location l has price $\eta(l)$, then slope associated with l is $-\eta(l)$ (see STEP 3 in Algorithm 1).

Let l_1, \dots, l_n be the successors of l_{min} , with cost functions f_1, \dots, f_n . Each of these cost functions are piecewise affine continuous over the domain $[0, 1]$. The first thing to do is to superimpose f_1, \dots, f_n , and obtain the cost function f corresponding to the interior of f_1, \dots, f_n (l_{min} is a player 1 location and would like to obtain the minimal cost, hence the interior). The line segments comprising f come from the various f_i . Let $dom(f) = [0, 1]$ be composed of $0 = u_{i_1} \leq v_{i_1} = u_{i_2} \leq \dots u_{i_m} \leq v_{i_m} = 1$: that is, $f(x) = f_{i_j}(x)$, $dom(f_{i_j}) = [u_{i_j}, v_{i_j}]$, for $i_j \in \{1, 2, \dots, n\}$ and $1 \leq j \leq m$. Let us denote f_{i_j} by g_j , for $1 \leq j \leq m$. Then, f is composed of g_1, g_2, \dots, g_m , and $dom(f)$ is composed of $dom(g_1), \dots, dom(g_m)$ from left to right. Let $dom(g_i) = [u_i, v_i]$. Step 2 of the algorithm achieves this.

For a given valuation $\nu(x)$, if l_{min} is an urgent location, then player 1 would go to a location l_k if the interior f is such that $f(\nu(x)) = g_k(\nu(x))$ (the least cost is given by g_k , obtained from the outside cost function of l_k). If l_{min} is not an urgent location, then player 1

Algorithm 1: Optimal Cost Algorithm when l_{min} is a Player 1 location

Let l_1, \dots, l_n be the successors of l_{min} with optcost functions $f_1, f_2 \dots f_n$;

STEP 1 : Superimpose : Superimpose all the optcost functions $f_1, f_2 \dots f_n$;

STEP 2 : Interior : Take the interior of the superimposition; call it f ;

Let f be composed of line segments $g_1, g_2 \dots g_m$ such that $g_i \in \{f_1, \dots, f_n\}$, for all i .

$\forall k$, let the domain of g_k be $[u_k, v_k]$. Set $i = m$;

STEP 3 : Selective Replacement : **while** $i \geq 1$ **do**

if $\text{slope of } g_i \leq -\eta(l_{min})$ **then**

replace g_i with line h_i with slope $-\eta(l_{min})$ and passing through $(v_i, g_i(v_i))$;

Let h_i intersect g_j (largest $j < i$) at some point $x = v_j'', v_j'' \in [u_j, v_j]$;

Update domain of g_j from $[u_j, v_j]$ to $[u_j, v_j'']$;

if $j < i - 1$ **then**

Remove functions g_{j+1} to g_{i-1} from f

Set $i = j$;

else

$i = i - 1$;

STEP 4 : Refresh Interior : Take the interior after STEP 3 and call it f' ;

if $l'' \rightarrow l_{min}$ **then**

update the optcost function of l''

would prefer delaying t units at l_{min} so that $\nu(x) + t \in [u_i, v_i]$ rather than goto some location l_i if $g_i(\nu(x)) > \eta(l_{min})(v_i - \nu(x))$. Again, g_i is a part of the outside cost function of l_i , and player 1 prefers delaying time at l_{min} rather than goto l_i since that minimizes the cost. In this case, the cost function f is refined by replacing the line segment g_i over $[u_i, v_i]$ by another line segment h_i passing through $(v_i, g_i(v_i))$, and having a slope $-\eta(l_{min})$. Step 3 of the algorithm does this.

Recall that by our transformation 2, the value of clock x in any player 1 location is $\leq 1 - \delta$. The value of x is in $[1 - \delta, 1 + \delta]$ only at a player 2 location $((A, e)_b^+)$ in the FRPTG, section 5.2). Hence, the domain of cost functions for player 1 locations is actually $[0, 1 - \delta]$, and not $[0, 1 + \delta]$. Let the domain of g_m be $[u_m, 1]$. Then we can split g_m into two functions g_m^1, g_m^2 with domains $[u_m, 1 - \delta]$ and $[1 - \delta, 1]$. Now, we ensure that no time is spent in the player 1 location l_{min} over $\text{dom}(g_m^2)$, by not applying step 3 of the algorithm for g_m^2 . This way, selective replacement of the cost functions g_i occur only in the domain $[0, 1 - \delta]$, and we remain faithful to transformation 2, and the semantics of RPTGs.

Computing Almost Optimal Strategies: The strategy corresponding to this computed optcost is derived as follows. f' is the optcost of location l_{min} computed in Step 4 of the algorithm. f' is composed of two kinds of functions (a) the functions g_i computed in step 2 as a result of the interior of superimposition and (b) functions h_i which replaced some functions g_j from f , corresponding to delay at l_{min} . For functions h_j of f' with domain $[u_j, v_j]$, we prescribe the strategy to delay at l_{min} till $x = v_j$ when entered with clock $x \in [u_j, v_j]$. For functions g_i , that come from f at Step 2, where g_i is part of some optcost function f_k , (f_k is the optcost function of one of the successors l_k of l_{min}), the strategy dictates moving immediately to l_k when entered with clock $x \in [u_i, v_i]$.

Termination: Finally, we prove the existence of a number N , the number of affine segments that appear in the cost functions of all locations. Start with the resetfree FRPTG with m locations having p segments in the outside cost functions. Let $\alpha(m, p)$ denote the total number of affine segments appearing in cost functions across all locations. The transformation of resetfree components $\mathcal{G}_{\mathcal{F}}$ into $\mathcal{G}_{\mathcal{F}}''$ gives rise to two smaller resetfree FRPTGs with $m - 1$ locations each, after separating out l_{min} . The resetfree FRPTG $(\mathcal{G}_{\mathcal{F}}, 1)$ with $m - 1$ locations indexed with 1 of the form $(l, 1)$ are solved first, these cost functions are added as outside cost functions to solve l_{min} , and finally, the cost function of l_{min} is added as an outside cost

function to solve the resetfree FRPTG $(\mathcal{G}_{\mathcal{F}}, 0)$ with $m-1$ locations indexed with 0 of the form $(l, 0)$. Taking into account the new sink target location added, we have $\leq p+1$ segments in outside cost functions in $(\mathcal{G}_{\mathcal{F}}, 1)$. This gives atmost $\beta = \alpha(m-1, p+1)$ segments in solving $(\mathcal{G}_{\mathcal{F}}, 1)$, and $\alpha(1, p+\beta) = \gamma$ segments to solve l_{min} , and finally $\alpha(m-1, p+\gamma)$ segments to solve $(\mathcal{G}_{\mathcal{F}}, 0)$. Solving this, one can easily check that $\alpha(m, p)$ is atmost triply exponential in the number of locations m of the resetfree component $\mathcal{G}_{\mathcal{F}}$. Obtaining a bound of the number of affine segments, it is easy to see that Algorithm 1 terminates; the time taken to compute almost optimal strategies and optcost functions is triply exponential.

We illustrate the computation of Optcost of a Player 1 location in Figure 2. The proof of Lemma 15 is given in Appendix G, while Lemma 16 follows from Lemma 15 and Step 4 of Algorithm 1.

► **Lemma 15.** *In Algorithm 1, if a function g_i (in f of Step 2) has domain $[u_i, v_i]$ and slope $\leq -\eta(l)$ then $\text{OptCost}(l, \nu) = (v_i - \nu) * \eta(l) + g(v_i)$.*

► **Lemma 16.** *The function f' in Algorithm 1 computes the optcost at any location l . That is, $\forall x \in [0, 1]$, $\text{OptCost}_G(l, x) = f'(x)$.*

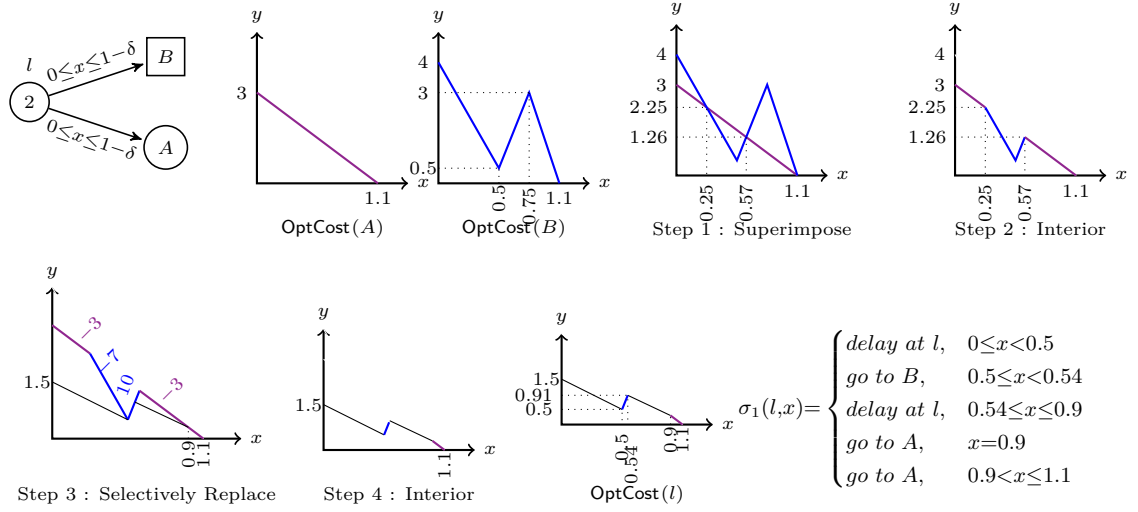
Note that the strategy under construction is a player 1 strategy, and player 1 has no control over the interval $[1, 1+\delta]$. $x \in [1, 1+\delta]$ after a positive perturbation, and is under player 2's control. Thus, at a player 1 location, proving for $x \in [0, 1]$ suffices.

5.4.2 l_{min} is a Player 2 location

If l_{min} is a player 2 location in the reset-free component $\mathcal{G}_{\mathcal{F}_i}$, then intuitively, player 2 would want to spend as little time as possible there. Keeping this in mind, we first run steps 1, 2 of Algorithm 1 by taking the exterior of f_1, \dots, f_n instead of the interior (player 2 would maximise the cost). There is no time elapse at l_{min} on running steps 1,2 of the algorithm. Let f be the computed exterior using steps 1,2. If f comprises of functions g_i having a greater slope than $-\eta(l)$, then it is better to delay at l_{min} to increase the cost. In this case, player 2 would want to improve his optcost using Step 3, by spending time at l_{min} . Finally, while doing Step 4, we take the exterior of the replaced functions h_i and old functions g_i . Recall that our transformations resulted in 3 kinds of player 2 locations : urgent, those with dwell-time restriction $[0, \delta]$ and finally those with $[\delta, 2\delta]$. The 3 cases are discussed in detail in Appendix H.

6 Conclusion and Future Work

In this paper we studied excess robust semantics and provided the first decidability result for excess semantics and improved the known undecidability result with 10 clocks to 5 clocks. To the best of our knowledge, the other known decidability result for robust timed games is under the conservative semantics for a fixed δ , [9]. As a consequence of our decidability result, the reachability problem for 1 clock PTG with arbitrary prices is shown to be decidable too under the assumption that the PTG does not have any negative cost cycle. The decidability we show is for a fixed perturbation bound $\delta > 0$. We use δ in the constraints of the dwell-time PTG after the first transformation for ease of understanding the robust semantics. Implementing this in step 3 of Algorithm 1 and ensuring no time elapse in the interval $[1-\delta, 1]$ takes no extra effort while l_{min} is a player 1 location. In that sense, we could have avoided explicit use of δ in the constraints in our simplifying transformations, and taken the appropriate steps in the algorithm itself. The existence of limit-strategy with $\delta \rightarrow 0$ seems rather hard. Our construction would not directly extend to limit-strategy problem as it is heavily dependant on the fixed δ .



■ **Figure 2** Optcost Computation for a Player 1 location ($\delta = 0.1$): we can keep the guards as $0 \leq x \leq 1$ and not apply Step 3 for $x \in [1 - \delta, 1]$.

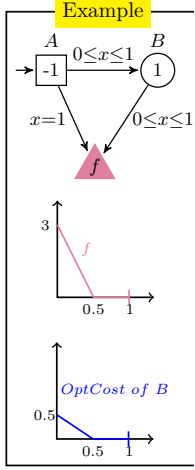
References

- 1 R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Proc. ICALP'04*, volume 3142 of *LNCS*, pages 122–133. Springer, 2004.
- 2 G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. W. Vaandrager. Minimum-cost reachability for priced timed automata. In M. D. Di Benedetto and A. L. Sangiovanni-Vincentelli, editors, *Proc. HSCC'01*, volume 2034 of *LNCS*, pages 147–161. Heidelberg, 2001. Springer.
- 3 P. Bouyer, T. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98:188–194, 2006.
- 4 P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In K. Lodaya and M. Mahajan, editors, *FSTTCS'04*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.
- 5 Patricia Bouyer, Kim Guldstrand Larsen, Nicolas Markey, and Jacob Illum Rasmussen. Almost optimal strategies in one clock priced timed games. In *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*, pages 345–356, 2006.
- 6 Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust reachability in timed automata: A game-based approach. In *Automata, Languages, and Programming*, volume 7392 of *Lecture Notes in Computer Science*, pages 128–140. Springer, 2012.
- 7 Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust weighted timed automata and games. In Víctor Braberman and Laurent Fribourg, editors, *Proceedings of the 11th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'13)*, volume 8053 of *Lecture Notes in Computer Science*, pages 31–46. Buenos Aires, Argentina, August 2013. Springer.
- 8 T. Brihaye, V. Bruyère, and J. Raskin. On optimal timed strategies. In P. Pettersson and W. Yi, editors, *Proc. FORMATS'05*, volume 3829 of *LNCS*, pages 49–64. Springer, 2005.
- 9 Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Timed parity games: Complexity and robustness. *Logical Methods in Computer Science*, 7(4), 2011.

- 10 T. D. Hansen, R. Ibsen-Jensen, and P. B. Miltersen. A faster algorithm for solving one-clock priced timed games. In PedroR. D'Argenio and Hernán Melgratti, editors, *CONCUR 2013 – Concurrency Theory*, volume 8052 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2013.
- 11 N. Markey. Robustness in real-time systems. In *Industrial Embedded Systems (SIES), 2011 6th IEEE International Symposium on*, pages 28–34, June 2011.
- 12 Michal Rutkowski. Two-player reachability-price games on single-clock timed automata. In *QAPL*, pages 31–46, 2011.
- 13 Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust controller synthesis in timed automata. In *CONCUR 2013 – Concurrency Theory*, volume 8052 of *Lecture Notes in Computer Science*, pages 546–560. Springer, 2013.

Appendix

A Cost Functions



We illustrate the cost functions with an example. In the PTG given here, the cost function f corresponding to the target gives the cost incurred when the target is entered various values of clock x . For example, if target is reached with clock value $x = 0$ then cost incurred is $cost = 3 = f(0)$ while $cost = 0$ if entered with $x \in [0.5, 1]$. Suppose B is entered with $x = 0$ and Player 1 decides to go to the target immediately with no delay at B (i.e; delay $d = 0$) then the cost is $cost = 1 * d + f(v) = 1 * 0 + f(0) = 3$, and $x = v$ upon entering the target. However, if player 1 waited at B till $x = 0.5$ and then went to target, the cost is $1 * 0.5 + f(0.5) = 0.5$. Similarly, if $d = 0.75$ then the cost is 0.75. From this, we can infer that the best strategy for Player 1 to achieve the optimal cost is to wait till $x = 0.5$ and then go to target. The second function labelled *OptCost of B* gives the optimal cost achievable for every value of x that B is entered with. Similar analysis for location A , reveals that the cost incurred is -1 if Player 2 went to target directly. Else, he could wait at A and then go to B . Due to the negative price at A , it is obvious that the best strategy for Player 2 is to go to B immediately. Thus, the optimal cost function for A is the same as that of B .

B UndecidabilityProof

We present below a set of figures which depict in full detail the simulation of all the instructions of two counter machine - increment, zero test and decrement.

First we describe a few *support modules* that will be used in the main modules for simulating increment, zero test and decrement instructions.

B.1 Prevent perturbation module

For correct simulation of the instructions, it will often be needed that the delay made by controller should not be perturbed by perturbator. The module in Figure 3 shows the construction that prevents perturbator from making any perturbation along the edge from A to B . In run ρ , the edge from B to the target ensures that if the delay chosen at A was

Module : Prevent perturbation

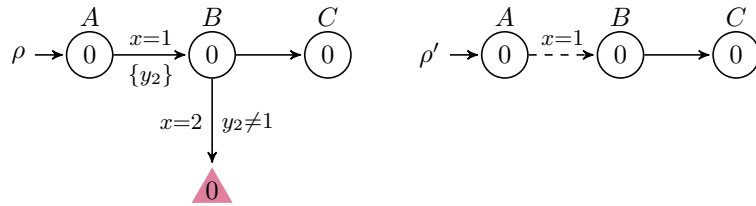
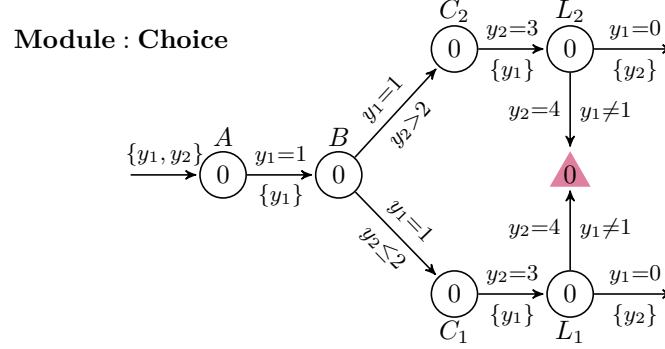


Figure 3 Prevent perturbation module: x is some clock and $x = k$ could be the constraint for any $k \in \mathbb{N}$. The triangle with 0 represents target location l with cost function $cf(l) : \mathbb{R}_{\geq 0} \rightarrow 0$.

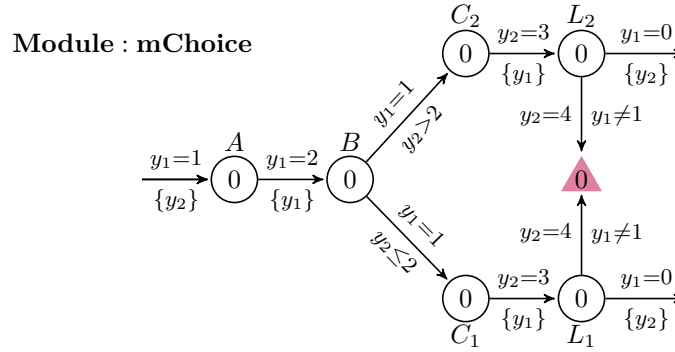
perturbed then controller can achieve a cost 0. For better readability, we represent these unperturbed edges as dashed arrows as shown in path ρ' . We note that the clock which is used in the equality constraint, (x in Figure 3) cannot be reset along the same edge. If we do not specify a clock that is being reset along the dashed edge, we consider it to be y_2 . For any other clock, we show it as being reset along the dashed edge. Note that in the ‘prevent perturbation module’, we need at least one equality constraint ($x = 1$ in Figure 3), thus ensuring a deterministic delay.

B.2 Choice module

Since we consider a priced timed automaton and not a PTG, perturbator does not own a location from where it can suggest the successor location of its choice. We show in Figure 4, the construction of a module that allows perturbator to choose the successor location. The



■ **Figure 4** Choice module : Perturbator can choose to go to C_2 if he perturbs the delay at B by a positive value. If he does not perturb or perturbs by a negative value then goes to C_1 .

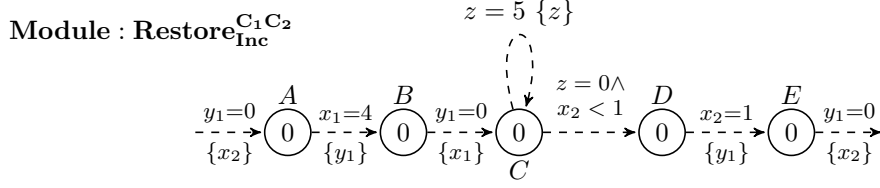


■ **Figure 5** mChoice module: The mChoice (modified choice) module is the same as the Choice module except for the fact that here the value of clock y_1 is 1 upon entry.

delay from location A to location B can be perturbed by perturbator. Controller chooses C_2 as the successor if the perturbation is positive, and chooses C_1 as its successor if the perturbation is negative. We note that if the module was entered with $x_1 = \alpha_1, z = \beta, x_2 = \alpha_2, y_1 = y_2 = 0$ then upon leaving either L_1 or L_2 the clock values are $x_1 = 3 + \alpha_1, z = 3 + \beta, x_2 = 3 + \alpha_2, y_1 = y_2 = 0$. The mChoice (modified choice) module shown in Figure 5 is the same as the Choice module except for the fact that here the value of clock y_1 is 1 upon entry. Thus the constraint on the edge between locations A and B is $y_1 = 2$ instead of $y_1 = 1$ as in choice module. Here also the value of clocks x_1, z and x_2 are increased by 3 as in the choice module while clocks y_1 and y_2 have value 0 on exit.

B.3 Restore module

Both choice and mChoice modules add a shift of 3 to the clock values x_1, x_2 and z . Since the main modules simulating increment and decrement of the counters expect the values to be in their normal forms, we need to remove the shift of 3; this is achieved by the Restore module



■ **Figure 6** Restore module : This is actually a group of four modules. $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ is shown in the figure.

shown in Figure 6. The restore modules used in the main modules simulating the operations on counter C_1 are a group of four different modules as mentioned below. $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ denotes the module used as part of the Increment module for counter C_1 . We also similarly have $\text{Restore}_{\text{Dec}}^{C_1 C_2}$ module which is used as part of the Decrement module. The $\text{Restore}_{\text{Dec}}^{C_1 C_2}$ module is similar to the $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ module with the only difference being that the clock constraint on the loop on C is $z = 6$ instead of $z = 5$ as in the $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ module. $C_1 C_2$ here denotes that the fractional part of clock x_1 is more than the fractional part of clock z . We also use $\text{Restore}_{\text{Dec}}^{C_2 C_1}$ and $\text{Restore}_{\text{Inc}}^{C_2 C_1}$ to denote that the fractional part of clock z is more than the fractional part of clock x_1 . The $\text{Restore}_{\text{Inc}}^{C_2 C_1}$ module can be obtained from $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ module by replacing all the occurrences of clock x_1 with clock z and replacing all the occurrences of clock z with clock x_1 . $\text{Restore}_{\text{Dec}}^{C_2 C_1}$ can also be obtained from $\text{Restore}_{\text{Dec}}^{C_1 C_2}$ in the same way. The edge from location C to location D forces controller to take the loop at location C only once. The $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ and $\text{Restore}_{\text{Inc}}^{C_2 C_1}$ modules are entered with clock values $x_1 = 3 + \frac{1}{2^i} + \varepsilon_1, z = 4 + \frac{1}{2^j} + \varepsilon_2, x_2 = y_1 = y_2 = 0$, at the starting location A of the module. At location E , the clock values are $x_1 = \frac{1}{2^i} + \varepsilon_1, z = \frac{1}{2^j} + \varepsilon_2, x_2 = y_1 = y_2 = 0$, i.e. restored to their normal form.

The restore modules used in the modules simulating operations on counter C_2 are analogous. Corresponding to $\text{Restore}_{\text{Inc}}^{C_1 C_2}$, the delays at locations A, C and D are $1 - \frac{1}{2^i} - \varepsilon_1, \frac{1}{2^i} + \varepsilon_1 - \frac{1}{2^j} - \varepsilon_2$ and $\frac{1}{2^j} + \varepsilon_2$ respectively, while the delays at locations B and E are 0. The value of clock z at the entry of the $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ and $\text{Restore}_{\text{Dec}}^{C_2 C_1}$ is $5 + \frac{1}{2^j} + \varepsilon_2$ and the clock values at the exit are as $\text{Restore}_{\text{Inc}}^{C_1 C_2}$ and $\text{Restore}_{\text{Inc}}^{C_2 C_1}$ modules.

We show below the main modules which are used for simulating zero test and decrement. We show here the modules corresponding to the operations on counter C_1 . The modules corresponding to the operations on counter C_2 are analogous.

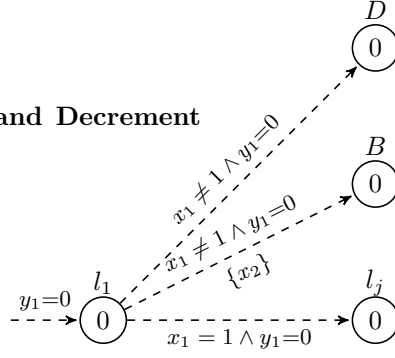
B.4 Decrement module

The module simulating decrement of counter C_1 is shown in Figure 7. Recall that by the normal form, the values of the clocks are $x_1 = \frac{1}{2^i} + \varepsilon_1, z = \frac{1}{2^j} + \varepsilon_2, y_1 = y_2 = x_2 = 0$ at l_1 .

1. Assume that $c_1 > 0$ at l_1 . Controller can choose to goto B or D , since the constraints on both the edges are the same. If $c_1 > 1$, controller chooses to goto B , and if $c_1 = 1$, then controller goes to D . Consider $c_1 > 1$, and controller visiting B . By the encoding, $x_1 = \frac{1}{2^i} + \varepsilon_1, i > 1, z = \frac{1}{2^j} + \varepsilon_2, x_2 = y_1 = y_2 = 0$. Here ε_1 and ε_2 denote errors accumulated so far in clocks x_1 and z due to perturbation made by perturbator so far. Figure 8 shows the section of the module shown in Figure 7 starting from location B . This section simulates the decrement of counter C_1 when the value of the counter is greater than 1. The value of clock z simulating counter C_2 remains unchanged.

Let us denote the value of x_1 at the entry of the module Decrement C_1 in Figure 8, i.e. $\frac{1}{2^i} + \varepsilon_1$ by α . Thus the delays at locations B and C are respectively $1 - \alpha$ and α . On entry at D , we thus have $x_2 = \alpha, y_1 = 0, y_2 = 1$. A non-deterministic time t is spent at

Module : Zero test and Decrement



■ **Figure 7** Zero test and Decrement Module: This module simulates the instruction **If** ($c_1 = 0$) **go to** l_j **else go to** l'_j . The extensions from B and D are shown in Figures 8 and 9 respectively.

D simulating the decrement of C_1 . Ideally, t must be $1 - 2\alpha$. Perturbator can perturb it by δ , where δ can be both positive and negative and clock x_1 is reset. On entering E we thus have $x_1 = 0, y_1 = t + \delta, x_2 = \alpha + t + \delta$. At the entry to mChoice module, the values of the clocks are $x_1 = 1 - t - \delta, z = 2 + \frac{1}{2j} + \varepsilon_2, x_2 = 1 + \alpha, y_1 = 1, y_2 = 0$. To correctly decrement C_1 (whose value is i), $1 - t$ should be exactly 2α , i.e. $\frac{1}{2^{i-1}} + 2\varepsilon_1$.

Perturbator uses the mChoice module to either continue the simulation (by going to the Restore module) or verifies controller's delay t . Due to the mChoice module, the clock values are $x_1 = 4 - t - \delta, z = 5 + \frac{1}{2j} + \varepsilon_2, x_2 = 4 + \alpha, y_1 = 0, y_2 = 0$. If perturbator chooses to continue the simulation, then the Restore module restores the clocks back to normal form and hence upon entering l'_j the clock values are $x_1 = 1 - t - \delta, z = \frac{1}{2j} + \varepsilon_2, x_2 = 0, y_1 = 0, y_2 = 0$. Thus, we have $x_1 = \frac{1}{2^{i-1}} + 2\varepsilon_1 - \delta$, where $2\varepsilon_1 - \delta$ is the value due to the perturbations so far.

However, if perturbator chooses to verify, he first goes to yet another Choice module. If $1 - t > 2\alpha$, then the module $Test Dec_{>}^{C_1}$ is used and if $1 - t < 2\alpha$, then the module $Test Dec_{<}^{C_1}$ is used. Note that due to the two Choice modules one after the other, the clock values upon entering $Test Dec_{<}^{C_1}$ or $Test Dec_{>}^{C_1}$ are $x_1 = 7 - t - \delta, x_2 = 7 + \alpha, y_1 = y_2 = 0$.

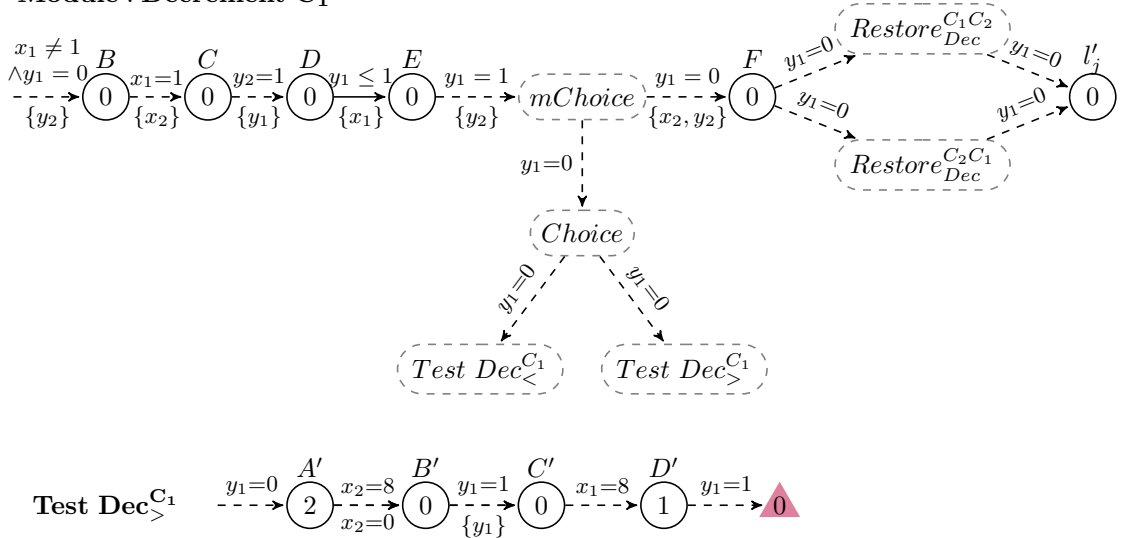
Test Dec $_{>}^{C_1}$: At A' , on entry we have $x_1 = 7 - t - \delta, x_2 = 7 + \alpha, y_1 = y_2 = 0$. A time $1 - \alpha$ is spent at A' with accumulated cost $2 - 2\alpha$.² On entry to B , we have $x_1 = 8 - \alpha - t - \delta, y_1 = 1 - \alpha$. A time α is spent at B' , and $x_1 = 8 - t - \delta$. A time $t + \delta$ is spent at C' , obtaining $y_1 = t + \delta$. A time $1 - t - \delta$ is spent at D' obtaining the accumulated cost $2 - 2\alpha + 1 - t - \delta$. The target is reached with this cost. If $1 - 2\alpha > t$, then this is $> 2 - \delta$. The perturbator can choose $\delta < 0$, making this cost > 2 .

2. Controller chooses the outgoing edge to D in Figure 7 if c_1 is 1 in which case the decremented value is 0 which is encoded by the exact value $x_1 = 1$. The module from D has been shown in Figure 9.

Figure 9 shows the section of the module of Figure 7 starting from location D . This section simulates the decrement of counter C_1 when $c_1 = 1$. Upon entering D , in the Test and Decrement module, the clock values are $x_1 = \frac{1}{2} + \varepsilon_1, z = \frac{1}{2j} + \varepsilon_2, x_2 = y_1 = y_2 = 0$. Let α denote the value of x_1 , i.e. $\frac{1}{2} + \varepsilon_1$. The time elapsed in locations D, E and F in Figure 9 are respectively $1 - \alpha, \alpha$ and 1. At the entry of the Choice module, the clock values are $x_1 = 1, z = 2 + \frac{1}{2j} + \varepsilon_2, x_2 = 1 + \alpha, y_1 = y_2 = 0$. Here x_1 encodes the counter value of C_1 exactly and perturbator cannot perturb the delay made by the controller.

Perturbator uses the Choice module to either continue the simulation or it can verify the

² The price 2 on A can be replaced with 1, by having a slightly longer sequence of transitions

Module : Decrement C_1 

■ **Figure 8** Decrement C_1 module : The section of the module shown in Figure 7 starting from location B . This section is used if $c_1 > 1$ before being decremented. It keeps the fractional part of clock z unchanged. The price 2 at A is a shorthand, and can be replaced with 1 on having a longer sequence of transitions.

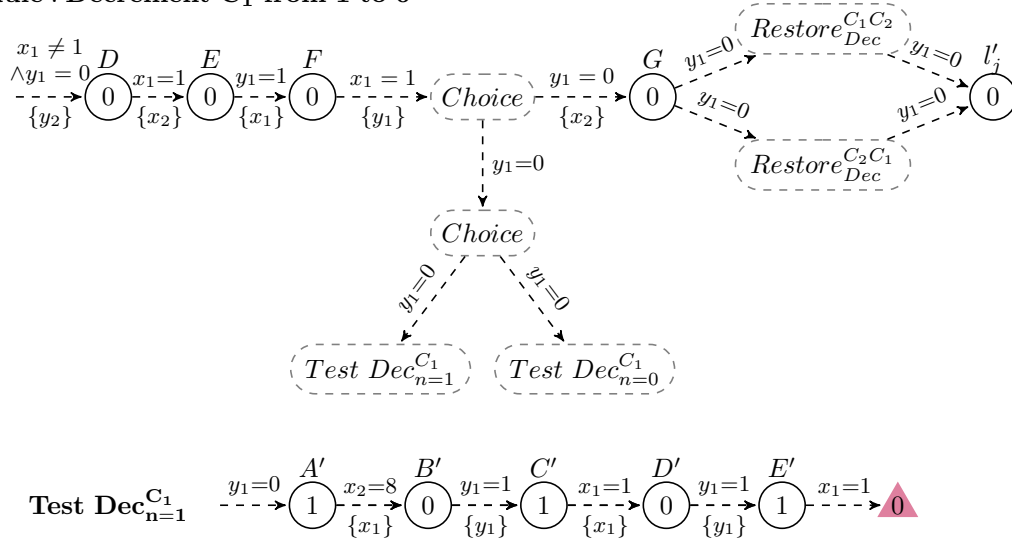
delay made by controller. Due to the Choice module, the clock values are $x_1 = 4, z = 5 + \frac{1}{2j} + \varepsilon_2, x_2 = 4 + \alpha, y_1 = y_2 = 0$. If perturbator chooses to continue the simulation then the Restore module restores the clocks back to the normal form and hence upon entering l'_j the clock values are $x_1 = 1, z_1 = \frac{1}{2j} + \varepsilon_2, x_2 = y_1 = y_2 = 0$.

However, if perturbator chooses to verify, he uses $Test Dec_{n=1}^{C_1}$ module to verify whether controller chose this branch (D) of the Test and Decrement module when $c_1 = 1$ or $c_1 > 1$. **Test Dec $_{n=1}^{C_1}$** : On entry, we have $x_1 = 7, z = 8 + \frac{1}{2j} + \varepsilon_2, x_2 = 7 + \alpha, y_1 = y_2 = 0$. The delays at locations are: at A' : $1 - \alpha$ obtaining $y_1 = 1 - \alpha$ on entering B' . A time elapse of α at B' gives $x_1 = \alpha$. Finally, at C' , we elapse $1 - \alpha$. Thus the cost incurred in this module is $3 - 3\alpha$. For $c_1 = 1$, this is $3 - \frac{3}{2} - 2\varepsilon_1 = 2 - \frac{1}{2} - 2\varepsilon_1$, and the minimum cost when $c_1 > 1$ is $3 - 3\frac{1}{2^2} - 2\varepsilon_1 = 2 + \frac{1}{4} - 2\varepsilon_1$. In the limit, as ε_1 tends to 0, the cost is ≤ 2 if controller chose the correct branch, that is, chose D when $c_1 = 1$.

3. Suppose controller chooses B instead of D when $c_1 = 1$. Then the value of clock x_1 after simulating the decrement operation will not be exact, i.e. will not be equal to 1. Now, if the next instruction involving controller C_1 is also a zero test and decrement operation, then controller will incorrectly move to l'_j instead of l_j while simulating this next zero test and decrement operation. For choosing B instead of D , controller will be punished while simulating this next zero test and decrement operation. Since the value of clock x_1 is not 1, while simulating this next zero test and decrement operation, controller will either go to B or D in the module in Figure 7.

- If B is chosen, t should equal $1 - 2\alpha$ for correct simulation. Now α being $1 + \varepsilon_1$, controller cannot delay for $1 - 2\alpha$ at location D of Figure 8 and hence is punished.
- If controller goes to location D in Figure 7, when $c_1 = 0$, then $x_1 = 1 + \varepsilon_1 = \alpha$ then perturbator moves to the module $Test Dec_{n=0}^{C_1}$. If $\varepsilon_1 > 0$, then the controller will get stuck in the transition from D to E (see Figure 9) and if $\varepsilon_1 < 0$, then the module $Test Dec_{n=0}^{C_1}$ in Figure 9 incurs a cost of $2 - 2\varepsilon_1 > 2$. The module $Test Dec_{n=0}^{C_1}$ can be drawn similar to $Test Dec_{n=1}^{C_1}$.

Module : Decrement C_1 from 1 to 0



■ **Figure 9** Decrement C_1 from 1 to 0 module : The section of the module shown in Figure 7 starting from location D . This section is used if $c_1 = 1$ before being decremented. It keeps the fractional part of clock z unchanged.

B.5 Complete Reduction

The entire reduction consists of constructing a module corresponding to each instruction I_i , $1 \leq i \leq n$, of the two-counter machine. The first location of the module corresponding to instruction I_1 is the initial location. We simulate the halting instruction I_n by a target location whose cost function assigns 2 to all clock values. We denote the robust timed automaton simulating the two counter machine by \mathcal{A} , s is the initial state $(l, \mathbf{0}, \mathbf{0})$.

► **Lemma 17.** *The two counter machine halts if and only if there is a strategy σ of controller such that $\limcost_{\mathcal{A}}(\sigma, s) \leq 2$.*

Proof. We first consider the case when the two counter machine halts. Suppose it halts in m steps. The cost incurred in m steps can be due to reaching one of the target states in a test module or reaching the *halt* instruction in m steps. We consider an ε such that $0 < 3^m \delta < \varepsilon$. In the first case, the cost is less than or equal to $2 + 2\varepsilon_b$, where by Lemma 18, $\varepsilon_b \leq \varepsilon$ and hence the cost is 2 in the limit. In the second case, controller simulates the two counter machine faithfully and reaches the target location corresponding to the halt instruction and hence the cost is 2 in the limit.

Now we consider the case when the two counter machine does not halt. Controller can simulate the two counter machine using the increment and the zero test and decrement modules corresponding to each of the instructions. The cost is ∞ if controller simulates the instructions faithfully and a target state is not reached. On the other hand, if controller makes an error, then it will be punished by perturbator in one of the test modules and cost will be non-zero. Hence the proof. ◀

Given an accumulated delay ε , the accumulated delay after one step due to the decrement and the increment modules are $2\varepsilon + \delta_1$ and $\varepsilon/2 + \delta_2$ respectively. The following lemma is from [7].

► **Lemma 18.** [7] *Consider the two functions $f : x \rightarrow 2x + 1$ and $g : x \rightarrow x/2 + 1$. For any $n \geq 1, x > 0$, and any $f_1, \dots, f_n \in \{f, g\}$, $f_1 \circ f_2 \circ \dots \circ f_n(x) \leq 3^n x$.*

We note that the prices used in all the modules in our reduction are only $\{0, 1\}$ and hence we have the undecidability result as given by Theorem 7.

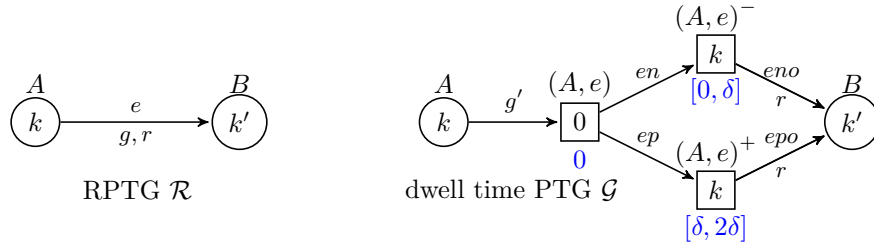
C Proof of Lemma 11

We first map the states of the RPTG \mathcal{R} and the dwell-time PTG \mathcal{G} . Let $\mathcal{S}(\mathcal{R})$ denote the set of states of the form (l, ν) as well as $((l, \nu), t, e)$ of the RPTG and $\mathcal{S}(\mathcal{G})$ denote the set of states of the dwell-time PTG.

► **Definition 19** (state map). We define a State Map $f : \mathcal{S}(\mathcal{R}) \rightarrow \mathcal{S}(\mathcal{G})$ as follows

- if l is a controller(perturbator) location then $f(l, \nu) = (l, \nu)$ as all controller locations of \mathcal{R} become player 1 locations in the dwell-time PTG \mathcal{G} , and all the perturbator locations of \mathcal{R} become player 2 locations in the dwell-time PTG \mathcal{G} ;
- Recall that the RPTG had states of the form $((l, \nu), t, e)$ corresponding to perturbator states (after controller chose a time delay and edge, perturbator decides the perturbation). Recall also that for every controller location l , and corresponding edge choice e made in the RPTG \mathcal{R} , we had the urgent player 2 location (l, e) immediately following the player 1 location l in the dwell-time-PTG \mathcal{G} constructed. That is, $f((l, \nu), t, e) = ((l, e), \nu + t - \delta)$

Note that $f(s)$ is a unique state in \mathcal{G} .



■ **Figure 10** Transitions of RPTG \mathcal{R} mapped to transitions in the constructed dwell-time PTG \mathcal{G}

► **Lemma 20.** *Given a path ρ in \mathcal{R} from s to s' , there exists a unique path ρ' in \mathcal{G} from $f(s)$ to $f(s')$. Additionally, $\text{Cost}(\rho) = \text{Cost}(\rho')$.*

The proof is quite straight forward and follows from the structure and the state map defined above.

Next, given a strategy σ_1 in \mathcal{R} , we shall define an equivalent strategy σ'_1 in \mathcal{G} in terms of the moves proposed. Let e be the edge from l to l' in \mathcal{R} . We map $\sigma(\rho.s) = (t, e)$ to $\sigma'(\rho'.f(s)) = (t', e')$ as follows

1. Controller strategy mapped to Player 1 strategy :

The strategy $\sigma_1(\rho.(l, \nu)) = (t, e)$ in \mathcal{R} leads to the state $((l, \nu), t, e)$. Corresponding to this, we have $\sigma'_1(\rho'.f(l, \nu)) = (t', e')$ such that $t' = t - \delta$ ³ and the player 1 location l moves into the urgent player 2 location (l, e) . This leads to $((l, e), \nu + t - \delta)$. e' is the edge in \mathcal{G} between l and (l, e) . Recall also that the time delay t in \mathcal{R} has been mapped to the time delay $t - \delta$ in the constructed PTG \mathcal{G} .

2. Perturbator strategy mapped to Player 2 strategy for perturbator locations:

A strategy $\sigma_2(\rho.(l, \nu)) = (t, e)$ in \mathcal{R} leads to $(l', \nu + t[r := 0])$. Correspondingly, we have in \mathcal{G} , $\sigma'_2(\rho'.(l, \nu)) = (t, e)$, giving the state $(l', \nu + t[r := 0])$ in \mathcal{G} .

³ $t \geq \delta$ in the \mathcal{R} due to robust semantics

3. Perturbator strategy mapped to Player 2 strategy for new locations :

Recall that we have $f((l, \nu), t, e) = ((l, e), \nu + t - \delta)$.

If we have the strategy $\sigma_2(\rho.((l, \nu), t, e)) = \epsilon \in [-\delta, \delta]$ in \mathcal{R} such that

- if $0 \leq \epsilon \leq \delta$ then $\sigma'_2(\rho'.((l, e), \nu + t - \delta)) = (0, ep)$ which results in $((l, e)^+, \nu + t - \delta)$ and $\sigma'_2(\rho'.((l, e)^+, \nu + t - \delta)) = (\delta + \epsilon, epo)$ resulting in $(l', \nu + t - \delta + \delta + \epsilon[r := 0])$.
- if $-\delta \leq \epsilon < 0$, let $\epsilon' = -\epsilon$ then $\sigma'_2(\rho'.((l, e), \nu + t - \delta)) = (0, en)$ which results in $\rho'.((l, e), \nu + t - \delta) \xrightarrow{0, en} ((l, e)^-, \nu + t - \delta)$ and $\sigma'_2(\rho'.((l, e)^-, \nu + t - \delta)) = (\delta - \epsilon', eno)$ resulting in $\rho'.((l, e)^-, \nu + t - \delta) \xrightarrow{\delta - \epsilon', eno} (l', \nu + t - \delta + \delta - \epsilon'[r := 0])$. Note that on entering $(l, e)^-$ with a value $\nu + t - \delta$, a time in $\epsilon \in [0, \delta]$ is spent at $(l, e)^-$, obtaining a valuation $\nu + t - \delta + \epsilon$. This corresponds to altering the time t spent by controller in \mathcal{R} to a value $t - \delta + \epsilon \in [t - \delta, t]$.

Similarly, given a strategy σ' in \mathcal{G} , we shall construct the equivalent strategy σ in \mathcal{R} as follows.

1. **Player 1 strategy to controller strategy** If $\sigma'_1(s)$ proposes a delay t then $\sigma_1(f^{(-1)}(s))$ proposes a delay $t + \delta$.
2. **Player 2 strategy to perturbator strategy in perturbator locations** If $\sigma'_2(s)$ proposes a delay t then $\sigma_2(f^{(-1)}(s))$ also proposes t .
3. **Player 2 strategy to perturbator strategy in controller locations** Suppose $\sigma'_2((l, e), \nu + t)$ suggests the path $((l, e)^+, \nu, c) \xrightarrow{\delta + \epsilon} (l', \nu')$. Then, $\sigma_2((l, \nu), t + \delta, e) \xrightarrow{\epsilon} (l', \nu')$.

► **Lemma 21.** *In the RPTG \mathcal{R} given in Figure 10, if g is $0 < x < 1$ then B is reached with $x \in [0, 1 + \delta]$. In the corresponding PTG \mathcal{G} too, B is reached with $x \in [0, 1 + \delta]$. We can establish the same for other possible guards too.*

► **Lemma 22.** *$\text{Cost}(s \xrightarrow{a} s') = \text{Cost}(f(s) \xrightarrow{a'} f(s'))$. That is, the cost of a transition from s to s' in the RPTG \mathcal{R} is the same as the cost of going from $f(s)$ to $f(s')$ in the dwell-time PTG \mathcal{G} . However, we need multiple transitions to reach from $f(s)$ to $f(s')$.*

Both the above lemmas follow from the definition of η' and the delays adjusted over l , $(l, e)^-$ and $(l, e)^+$ in the PTG \mathcal{G} .

► **Lemma 23.** *Given a strategy σ_1 in \mathcal{R} and the corresponding strategy σ'_1 in \mathcal{G} , for every state s in \mathcal{R} , $\text{Cost}(s, \sigma_1) = \text{Cost}(f(s), \sigma'_1)$.*

Proof. Recall that $\text{Cost}_{\mathcal{R}}(s, \sigma_1) = \sup_{\sigma_2 \in \text{Strat}_2(\mathcal{R})} (\text{Cost}_{\mathcal{R}}(\text{Outcome}(s, \sigma_1, \sigma_2)))$.

Part 1: $\text{Cost}_{\mathcal{R}}(s, \sigma_1) \leq \text{Cost}_{\mathcal{G}}(f(s), \sigma'_1)$

Consider a strategy σ_2 in \mathcal{R} . We can construct a strategy σ'_2 in \mathcal{G} as outlined above. From Lemma 22, it is clear that the $\text{Cost}_{\mathcal{R}}(\text{Outcome}(s, \sigma_1, \sigma_2)) \leq \text{Cost}_{\mathcal{G}}(\text{Outcome}(f(s), \sigma'_1, \sigma'_2))$.

Part 2: $\text{Cost}_{\mathcal{G}}(f(s), \sigma'_1) \leq \text{Cost}_{\mathcal{R}}(s, \sigma_1)$

Consider a strategy σ'_2 in \mathcal{G} . We can construct a strategy σ_2 in \mathcal{R} as outlined above. The selected semantics of \mathcal{G} and Lemma 21 ensure that all of σ'_2 proposed delays can be emulated in \mathcal{R} too. ◀

Along the same lines as the lemma above, we could also prove that $\text{Cost}(s, \sigma_2) = \text{Cost}(f(s), \sigma'_2)$. These two results pave the way for relating the optimal costs for states in the two games. We shall establish $\text{OptCost}_{\mathcal{R}}^{\delta}(s) = \text{OptCost}_{\mathcal{G}}^{\delta}(f(s))$ by proving two inequalities

(1) $\text{OptCost}_{\mathcal{R}}^{\delta}(s) \leq \text{OptCost}_{\mathcal{G}}^{\delta}(f(s))$ and

(2) $\text{OptCost}_{\mathcal{G}}^{\delta}(f(s)) \leq \text{OptCost}_{\mathcal{R}}^{\delta}(s)$

$\text{OptCost}_{\mathcal{R}}^{\delta}(s) \leq \text{OptCost}_{\mathcal{G}}(f(s))$

Consider a strategy σ_1 in \mathcal{R} and construct an equivalent strategy σ'_1 in \mathcal{G} (this is possible, Lemma 20). Now we shall prove that

$$\sup_{\sigma_2 \in \text{Strat}_2(\mathcal{R})} (\text{Cost}(\text{Outcome}(s, \sigma_1, \sigma_2))) = \sup_{\sigma'_2 \in \text{Strat}_2(\mathcal{G})} (\text{Cost}(\text{Outcome}(f(s), \sigma'_1, \sigma'_2))).$$

To this end, let us consider a perturbator strategy σ_2 in \mathcal{R} . Then we can construct an equivalent Player 2 strategy σ'_2 such that $\text{Cost}(\text{Outcome}(s, \sigma_1, \sigma_2)) = \text{Cost}(\text{Outcome}(s, \sigma'_1, \sigma'_2))$ (follows from Lemma 22). Thus, we have shown that the set of strategies in \mathcal{G} is at least as large as those in \mathcal{R} and whatever costs are achieved in \mathcal{R} can be achieved in \mathcal{G} too.

$\text{OptCost}_{\mathcal{G}}(s) \leq \text{OptCost}_{\mathcal{R}}^{\delta}(s)$

We shall now construct strategies in \mathcal{R} from strategies in \mathcal{G} . If $\sigma'_1(s)$ proposes a delay t then $\sigma_1(f^{(-1)}(s))$ proposes $t + \delta$. Lemma 21 ensures that $t + \delta$ will satisfy the guard. For example, if the guard was $0 < x < 1$ in \mathcal{R} then the delay chosen by σ'_1 is $< 1 - \nu(x) - \delta$.

Similarly, we construct strategy σ_2 from σ'_2 as specified above. If $\sigma'_2((l, e), \nu + t)$ suggests the path $((l, e)^+, \nu) \xrightarrow{\delta + \epsilon} (l', \nu')$. Then, $\sigma_2((l, \nu), t + \delta, e) \xrightarrow{\epsilon} (l', \nu'')$. We know that, $\nu'' = \nu'$. Once again, Lemma 21 ensures that if ν' is in an interval I then $\nu'' \in I$. For example, for the guard $0 < x < 1$, $\nu', \nu'' \in [0, 1 + \delta]$.

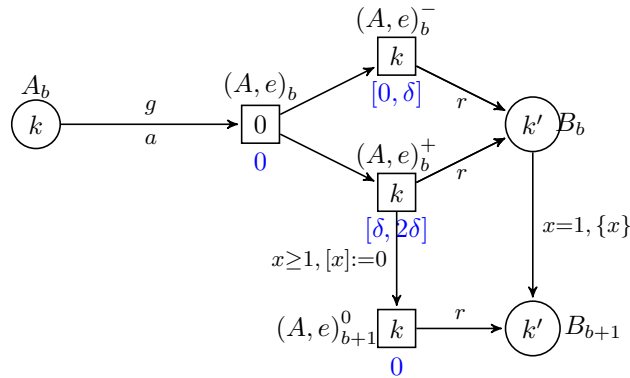
Once we have mapped the strategies, the proof of $\text{OptCost}_{\mathcal{G}}(s) \leq \text{OptCost}_{\mathcal{R}}^{\delta}(s)$ is along the same lines as the previous case.

► **Lemma 24.** *if σ_1 in \mathcal{R} is (ϵ, N) -acceptable then σ'_1 in \mathcal{G} is also (ϵ, N) -acceptable.*

A strategy in \mathcal{R} is said to be (ϵ, N) -acceptable if (1) it is memoryless, (2) is ϵ -optimal for every state and (3) partitions $[0, 1 + \delta]$ into at most N intervals.

From the definition of equivalent strategy σ'_1 , it is easy to see that if σ_1 is memoryless then so is σ'_1 . Additionally, if σ_1 has n intervals then σ'_1 would also have n intervals except that the intervals' end points would be shifted by δ as the delay prescribed by σ'_1 are $t - \delta$ when σ_1 suggests t . Finally, we can claim that ϵ -optimality is preserved from Lemma 23.

D Dwell time PTG to Dwell time FRPTG



■ **Figure 11** FRPTG

We have already defined in section 5.2, the mapping between the states of the dwell-time PTG \mathcal{G} and the constructed dwell-time FRPTG $\mathcal{G}_{\mathcal{F}}$. The mapping is defined in such a way

that a state (l, ν) in \mathcal{G} is mapped to the state $(l_b, \nu - b)$, whenever $\nu \in [b, b + 1]$. the integral part of the clock valuation is remembered in the state itself, while the valuation always stays in $[0, 1]$. The only exception to this is the location $(l, e)_b^+$, where the clock valuation can go up to $1 + \delta$. The state $((l, e)_b^+, \nu)$ in $\mathcal{G}_{\mathcal{F}}$ is the mapping of the state $((l, e)^+, b + \nu)$ in \mathcal{G} .

► **Lemma 25.** *Given a path ρ in \mathcal{G} from s to s' , there exists a unique path ρ' in $\mathcal{G}_{\mathcal{F}}$ from $f(s)$ to $f(s')$. Additionally, $\text{Cost}(\rho) = \text{Cost}(\rho')$.*

The proof of Lemma 25 is straightforward, given the mapping f . Any time elapse of 1 in any one state (l, ν) in \mathcal{G} is captured by starting from some $(l_b, \nu - b)$, and moving to $(l_{b+1}, (\nu + 1) - (b + 1))$ in $\mathcal{G}_{\mathcal{F}}$ and so on. Whenever the clock value reaches an integral value in \mathcal{G} , correspondingly in the $\mathcal{G}_{\mathcal{F}}$, the state is updated by remembering the new integral part, and updating the clock valuation to 0. Every path in \mathcal{G} corresponds to a path in $\mathcal{G}_{\mathcal{F}}$, where the constraints on the path are shifted by an appropriate integer, depending on the integral value remembered in the current state.

This also gives a mapping between the strategies of \mathcal{G} and $\mathcal{G}_{\mathcal{F}}$. Also, the costs are preserved across paths : any path in \mathcal{G} is mapped to a longer path in $\mathcal{G}_{\mathcal{F}}$ so that the individual time delays in $\mathcal{G}_{\mathcal{F}}$ never exceed 1. Since the prices of states are preserved by the mapping, the costs will add up to be the same. It is easy to see that a *copy-cat* strategy works between \mathcal{G} and $\mathcal{G}_{\mathcal{F}}$, and hence, costs, optimal costs are preserved. Since strategies are copy-cat, all properties like (ϵ, N) -acceptability are also preserved across games.

E FRPTG to Reset-free FRPTG

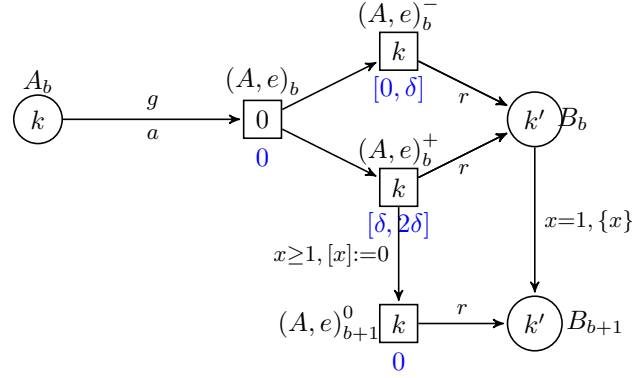
Given a one clock FRPTG $\mathcal{G}_{\mathcal{F}} = (L_1, L_2, \{x\}, X, \eta, T)$ with n resets (including fractional resets), we define a reset-free FRPTG as follows : $\mathcal{G}_{\mathcal{F}}' = (L'_1, L'_2, \{x\}, X', \eta', T')$ where

- For $l \in L_1$ and $0 \leq j < n$, we have $l^j \in L'_1$;
- For $l \in L_2$ and $0 \leq j < n$, we have $l^j \in L'_2$;
- $S \notin L_1 \cup L_2$ is a sink location such that $S \in L'_2$;
- X' has the following transitions.
 - $l^j \xrightarrow{g} l'^j$ if $l \xrightarrow{g} l' \in X$;
 - $l^j \xrightarrow{g, r} l'^{j+1}$ if $l \xrightarrow{g, r} l' \in X$, $j < n$ and r is either $\{x\}$ or $[x] := 0$;
 - $l^n \xrightarrow{g, r} S$ if $l \xrightarrow{g, r} l' \in X$ and r is either $\{x\}$ or $[x] := 0$;
 - $S \rightarrow S$;
- $\eta'(l^j) = \eta(l)$ and $l^j \in T'$ if $l \in T$.

We illustrate the construction of a resetfree FRPTG in Figure 13, corresponding to the FRPTG in Figure 12. Note that the locations in the upper rectangle form the the first copy $\mathcal{G}_{\mathcal{F}}-0$ and while the lower rectangle forms the second copy $\mathcal{G}_{\mathcal{F}}-1$. A copy $\mathcal{G}_{\mathcal{F}}-i$ indicates the number of resets seen so far from the initial location l_0^0 of the first copy $\mathcal{G}_{\mathcal{F}}-0$.

E.1 Proof of Lemma 13

Proof. Consider any state (l, ν) in $\mathcal{G}_{\mathcal{F}}$. The reduction from the FRPTG $\mathcal{G}_{\mathcal{F}}$ to the reset-free FRPTG $\mathcal{G}_{\mathcal{F}}'$ creates a new component (or copy) for each new reset, including fractional resets. Given that there are a total of n resets in the FRPTG, $\mathcal{G}_{\mathcal{F}}$, $n + 1$ reset-free components are created in the reset-free FRPTG $\mathcal{G}_{\mathcal{F}}'$, and the last component goes to a location with cost $+\infty$. By assumption, the cycles in each reset-free component are non-negative. Any cycle in the FRPTG $\mathcal{G}_{\mathcal{F}}$ involving a reset is mapped to a path in the reset-free FRPTG $\mathcal{G}_{\mathcal{F}}'$ ending at the location S with cost $+\infty$, while any reset-free cycle in $\mathcal{G}_{\mathcal{F}}$ is mapped to a cycle in one of the $n + 1$ reset-free components of the reset-free FRPTG $\mathcal{G}_{\mathcal{F}}'$. Clearly, for every strategy σ of player 1, 2 in $\mathcal{G}_{\mathcal{F}}$, there is a corresponding strategy σ' in the $\mathcal{G}_{\mathcal{F}}'$ and vice-versa, obtained using the above mapping of paths between $\mathcal{G}_{\mathcal{F}}$ and $\mathcal{G}_{\mathcal{F}}'$. Given that the prices of locations are



■ **Figure 12** FRPTG

preserved between $\mathcal{G}_{\mathcal{F}}$ and $\mathcal{G}_{\mathcal{F}'}$, the optimal cost from (l, ν) in $\mathcal{G}_{\mathcal{F}}$ is the same as the optimal cost from (l^0, ν) in $\mathcal{G}_{\mathcal{F}'}$.

Consider a (ϵ, N) -acceptable strategy σ' in $\mathcal{G}_{\mathcal{F}'}$. Consider a winning state (l^0, ν) . Let i be the minimum number of resets from state (l^0, ν) along any path compatible with σ' . That is, the player can win from (l^{n-i}, ν) but not from (l^{n-i+1}, ν) . If (l, ν) is not winning then we take $i = n + 1$. We denote by σ'_{n-i} the suggestions made by σ' in the $n - i^{\text{th}}$ copy in $\mathcal{G}_{\mathcal{F}'}$. We then assign $\sigma(l, \nu) = \sigma'_{n-i}(l^{n-i}, \nu)$. Thus, we obtain that $\text{Cost}_{\mathcal{G}_{\mathcal{F}}}((l, \nu), \sigma) = \text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^{n-i}, \nu), \sigma')$. Since (l^0, ν) and (l^{n-i}, ν) have the same outgoing transitions, we know that the strategy σ' from (l^0, ν) will be atleast as costly as $\text{OptCost}_{\mathcal{G}_{\mathcal{F}'}}(l^{n-i}, \nu)$. That is,

$$\text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^0, \nu), \sigma') \geq \text{OptCost}_{\mathcal{G}_{\mathcal{F}'}}(l^{n-i}, \nu) \quad (1)$$

Now, if $\text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^{n-i}, \nu), \sigma') > \text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^0, \nu), \sigma') + \epsilon$, then by Equation 1 we have $\text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^{n-i}, \nu), \sigma') > \text{OptCost}_{\mathcal{G}_{\mathcal{F}'}}(l^{n-i}, \nu) + \epsilon$ which means σ' is not ϵ -optimal. Thus we have,

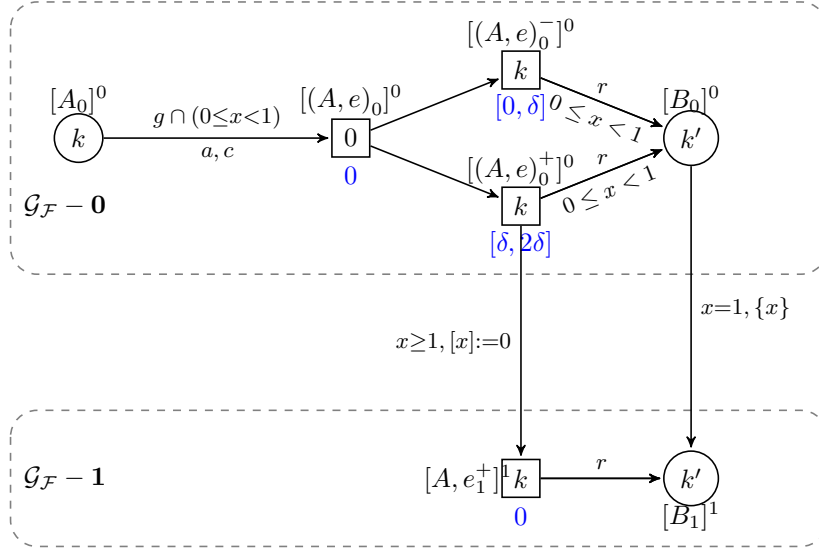
$$\text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^{n-i}, \nu), \sigma') \leq \text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^0, \nu), \sigma') + \epsilon$$

$$\text{Cost}_{\mathcal{G}_{\mathcal{F}}}((l, \nu), \sigma) = \text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^{n-i}, \nu), \sigma') \begin{cases} \leq \text{Cost}_{\mathcal{G}_{\mathcal{F}'}}((l^0, \nu), \sigma') + \epsilon \\ \leq \text{OptCost}_{\mathcal{G}_{\mathcal{F}'}}(l^0, \nu) + \epsilon + \epsilon \\ \leq \text{OptCost}_{\mathcal{G}_{\mathcal{F}}}(l, \nu) + 2\epsilon \end{cases}$$

◀

We shall now focus on informally explaining why fractional resets would not cause a problem. In a PTG without fractional resets, a resetting transition e (say $l \xrightarrow{x=1, x:=0} m$) taken twice takes us back to the same state $(m, x=0)$ twice. This crucial property is the back bone of the transformation which removes resets in [5]. The correctness proof is by constructing optcost preserving strategies for Player 1 in both \mathcal{G} and its resetfree equivalent \mathcal{G}' . Given a winning strategy for Player 1 in PTG \mathcal{G} , a strategy in \mathcal{G}' is constructed so as to ensure each resetting transition is taken atleast once. This is possible because a resetting transition e appearing the second time, results in the same state $(m, 0)$ and hence the transitions possible (and the optcost achievable) from the second resultant state $(m, 0)$ can be applied the first time this state occurs itself. In other words, the second occurrences of the transition are replacable as they result in the same unique state $(m, 0)$. It should be the case that a path exists such as to avoid the second occurrence of the resetting transition as the strategy is winning for Player 1.

Now, a similar reasoning will not work for fractional resets e' (say $l' \xrightarrow{x \geq 1, [x] := 0} m'$) as the resulting state (m', x) after a fractional reset transition is not unique (as the clock $x \in [0, \delta]$)



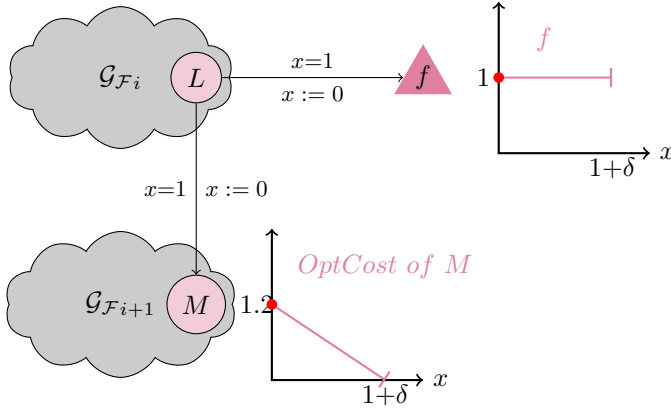
Resetfree FRPTG

■ **Figure 13** Resetfree FRPTG - two copies $\mathcal{G}_F - 0$ and $\mathcal{G}_F - 1$ corresponding to the number of resets encountered so far i.e; $\mathcal{G}_F - 0$ indicates that 0 resets have been seen so far and $\mathcal{G}_F - 1$ indicates 1 (fractional) reset has been seen.

and thus we can not adopt this argument directly. Firstly, the player 2 location $(l, e)_i^+$ is entered with $x \leq 1 - \delta$ (see Transformation 2) and a delay d makes $x \in [1, 1 + \delta]$. This delay happens entirely in this location and is chosen entirely by Player 2. From $(l, e)_i^+$, if player 2 moves to a $(l, e)_{i+1}^0$ location, then the value of x is in $[0, \delta]$. Note that the value of x , say ζ in $(l, e)_{i+1}^0$ is indeed the perturbation that happened in the RPTG \mathcal{R} : in the FRPTG, at $(l, e)_i^+$, player 2 elapses $\delta + \zeta$. Recall that if in \mathcal{R} , a location l was entered with value of x being ν , then in the FRPTG, we enter $(l, e)_i$ and $(l, e)_i^+$ with $\nu - i - \delta$. Player 2 at $(l, e)_i^+$ makes this value to be $\nu - i + \zeta$, which is exactly same as the perturbed value of x in \mathcal{R} when perturbator chooses a positive perturbation. The point to note is that whenever player 2 returns to $(l, e)_i^+$, the control of perturbation is his; thus, any ζ that is achieved the k th time can be achieved the first time itself by player 2. Moreover, if player 2 has a strategy to revisit $(l, e)_i^+$, then clearly, player 1 will lose, since after $n + 1$ times, the control reaches the target with cost ∞ . Note also that in Algorithm 1, while we solve for $(l, e)_i^+$, we will have the optcost function computed for $(l, e)_{i+1}^0$. Player 2 will choose to delay $\delta + \zeta$ for that ζ where the cost is maximal in the optcost of $(l, e)_{i+1}^0$.

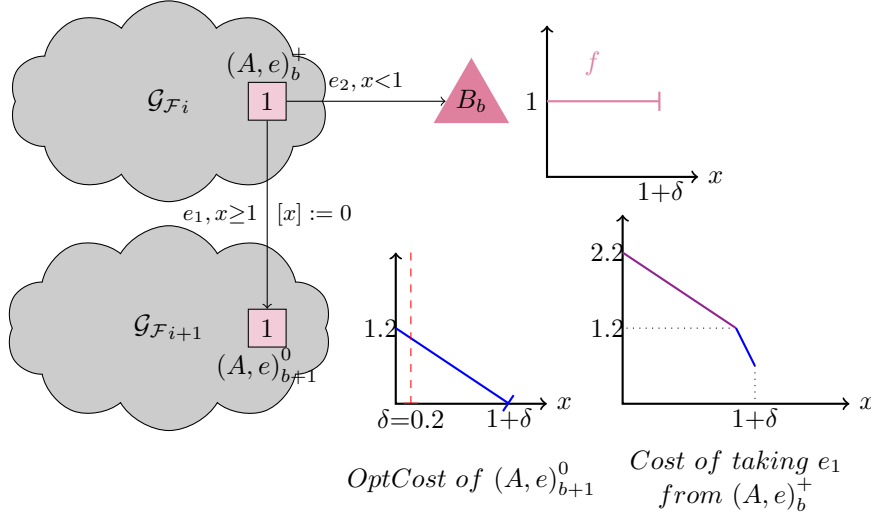
F Example : Solve Reset-Free FRPTG

We shall first look at how normal resets are handled. As detailed by the resetfree construction, each copy of the FRPTG is an SCC and there are $n + 1$ copies when the FRPTG has n resets. The $i + 1$ th copy is solved and its optcost functions are used as outside cost functions while solving the i th copy. This is depicted clearly in the figure below.



Location L in \mathcal{G}_{F_i} has two transitions with resets - one to a target and another to a location M in $\mathcal{G}_{F_{i+1}}$ whose optcost function has already been computed. Due to the clock reset, the target or M are entered with clock value $x = 0$ and hence the only values of interest are : the cost function f of the target at $x = 0$ i.e; $f(0) = 1$ and optcost for location M at $x = 0$ i.e; 1.2. Since L is a Player 1 location, the lower of these two values is picked and the corresponding transition is selected.

Now let us now consider fractional resets. The following figure depicts the previously considered example with normal reset replaced with fractional reset.



Recall from Transformation 2 (Dwell-time PTG \mathcal{G} to FRPTG $\mathcal{G}_{\mathcal{F}}$) that fractional resets occur only along transitions from $(A, e)_b^+$ to $(A, e)_{b+1}^0$. Lets call this transition e_1 . From the construction of FRPTG, we also know that the only other transition possible from $(A, e)_b^+$ is to location B_b , corresponding to the transition from A to B in the RPTG. Let us denote this transition as e_2 . by construction of the reset-free FRPTG, the constraint on $(A, e)_b^+ \rightarrow B_b$ is $x < 1$. Figuring out which part of the cost functions of B_b and $(A, e)_{b+1}^0$ to consider for the optcost computation of $(A, e)_b^+$ is a little different from the normal reset case. Here the guards on transitions can be considered as $0 \leq x < 1$ for e_2 and $1 \leq x \leq 1 + \delta$ for e_1 .

Hence we should consider the entire cost function of B_b , while taking only the $x \in [0, \delta]$ part from the function of $(A, e)_{b+1}^0$. Recall that fractional resets removed the integer part of x , thereby taking x from $[1, 1 + \delta]$ to $[0, \delta]$. Thus the cost function of taking the transition to $(A, e)_{b+1}^0$ is equal to (delay of waiting at $(A, e)_b^+$ till $x = v \in [1, 1 + \delta]$) + (optcost of $(A, e)_{b+1}^0$

at $1 - v$). We compute this cost as outlined in Figure 15. It is easy to see that since the price of $(A, e)_b^+$ is 1, and the slope of the optcost function of $(A, e)_{b+1}^0$ is -1.2 , it is more profitable to reach $(A, e)_{b+1}^0$ at $x = 0$ than at $x = \delta$ i.e; the transition e_1 when $x = 1$, thus reaching $(A, e)_{b+1}^0$ at $x = 0$ after the fractional reset, rather than wait at $(A, e)_b^+$ till $x = 1 + \delta$ and then reach $(A, e)_{b+1}^0$ at $x = \delta$ yielding only 2.16 (wait till $x = 1 + \delta$ incurring 1.2 and then optcost of $(A, e)_{b+1}^0$ at $x = \delta$ is 0.96).

We consider this cost function of taking the transition e_1 and the cost function of B_b to compute the optcost function of $(A, e)_b^+$. In this example, it is clearly better for Player 2 to take e_1 at $x = 1$ and hence the cost function of taking e_1 is the optcost of $(A, e)_b^+$.

G Algorithm for OptCost Computation : l is a player 1 location

We first prove Lemma 15.

Proof. The optcost computation for a location l_{min} is done using the already computed optcosts of all successors of l_{min} , which we now treat as outside cost functions. The Steps 1 and 2 in Algorithm 1, superimpose the outside cost functions corresponding to l_{min} and take the interior. Recall that step 3 is applied right to left : we start the selective replacement from $\nu = 1 + \delta$ and proceed towards 0. We know that up to v_i , for all $\nu \geq v_i$, $\text{OptCost}(l_{min}, \nu) = f(v_i)$.

Now we have to compute the optcost for $\nu \in [u_i, v_i]$. As we have taken the interior of the superimposed function in Step 2, we know that g_i is the best (lowest) possible cost if we do not delay at l_{min} . Let us determine if delaying at l_{min} is more profitable than following g_i . The two options we have are :

1. Follow g_i whose slope is $-m$. The line segment g_i is given by $y = -mx + c$ where $c = f(v_i) + mv_i$, since $y = g_i(v_i) = f(v_i)$ at $x = v_i$; (f is continuous, and is composed of g_1, \dots, g_m) and
2. delay at l_{min} till $x = v_i$ and exit at v_i . The line segment corresponding to the delay at l_{min} is $y = -\eta(l_{min}) * x + c'$ where $c' = f(v_i) + \eta(l_{min}) * v_i$ as we delay at l_{min} until $x = v_i$ and follow f , thus obtaining $f(v_i)$ at $x = v_i$

Now comparing these two equations we get the following.

$$\begin{aligned} -mx + c &\sim -\eta(l_{min}) * x + c' \\ -mx + f(v_i) + mv_i &\sim -\eta(l_{min}) * x + f(v_i) + \eta(l_{min}) * v_i \\ -mx + mv_i &\sim -\eta(l_{min}) * x + \eta * v_i \\ m * (v_i - x) &\sim \eta(l_{min}) * (v_i - x) \end{aligned}$$

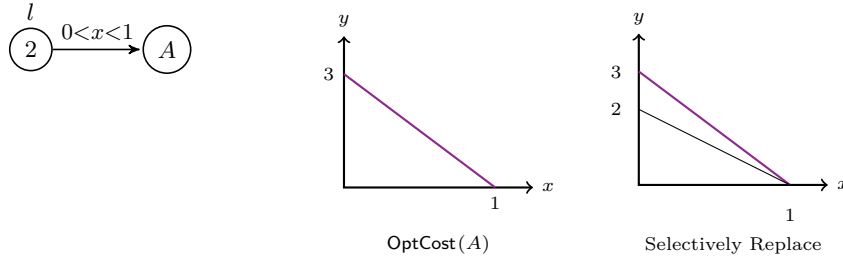
If $x \leq v_i$ and $\eta(l_{min}) \leq m$, then we conclude \sim is \geq

Thus, we observe that delaying at l_{min} is better. The above discussion is for Player 1 but can be easily adapted to Player 2. In a similar fashion, we can argue that delaying at l_{min} till $x \leq v'_i < v_i$ is worse than delaying till $x \leq v_i$ i.e; Player 1 prefers to wait until v_i instead of exiting and following g_i at some point $v'_i < v_i$. ◀

G.1 OptCost Computation for All Constraints

In the computation in Algorithm 1, we have assumed that all the transitions from $l \xrightarrow{e} l'$ have a guard $0 \leq x \leq 1$. We shall now illustrate how to compute optcost of l if the guards on the outgoing transitions are different.

While optimal strategies are possible with closed constraints, it is known that optimal strategies need not exist with open constraints.



■ **Figure 14** Optcost Computation for guard $0 < x < 1$

Constraints on all the outgoing edges are $0 < x < 1$

We shall illustrate how to obtain ϵ -optimal strategies with open constraints. Consider the Figure 14. Here the guard is $0 < x < 1$ and clearly the $\text{OptCost}_{\mathcal{G}}(l, 0) = 2$ and there is no strategy to achieve that. Hence we want to find the ϵ -optimal strategy achieving $< 2 + \epsilon$. Pick $t = \frac{\epsilon}{m_{max}+1}$ where m_{max} is the slope with the largest absolute value seen among the outside cost functions. Here $m_{max} = 3$ ($\text{OptCost}_{(A)}$ is $y = -3x + 3$). Let $\epsilon = 0.1$. Then $t = 0.025$. Now let's fix the strategy to wait at l till $x < 1 - t$ and go to A at $x = 1 - t^4$. Then $\text{OptCost}_{\mathcal{G}}(l, 0) = 2 * (1 - t) + f(1 - t)$ where f given by $y = -3x + 3$ is the optcost function of A . Thus $\text{OptCost}_{\mathcal{G}}(l, 0, 0) = 2.025 < 2 + 0.1$. Extending this to several successors of l is simple and follows all the steps of Algorithm 1. At $1 - t$, take the transition to the location prescribed by f' in Step 4. Note that this method would work for $0 < x < 1 - \delta$ by simply replacing 1 with $1 - \delta$ in the discussion above.

Constraint $1 - \delta < x < 1$

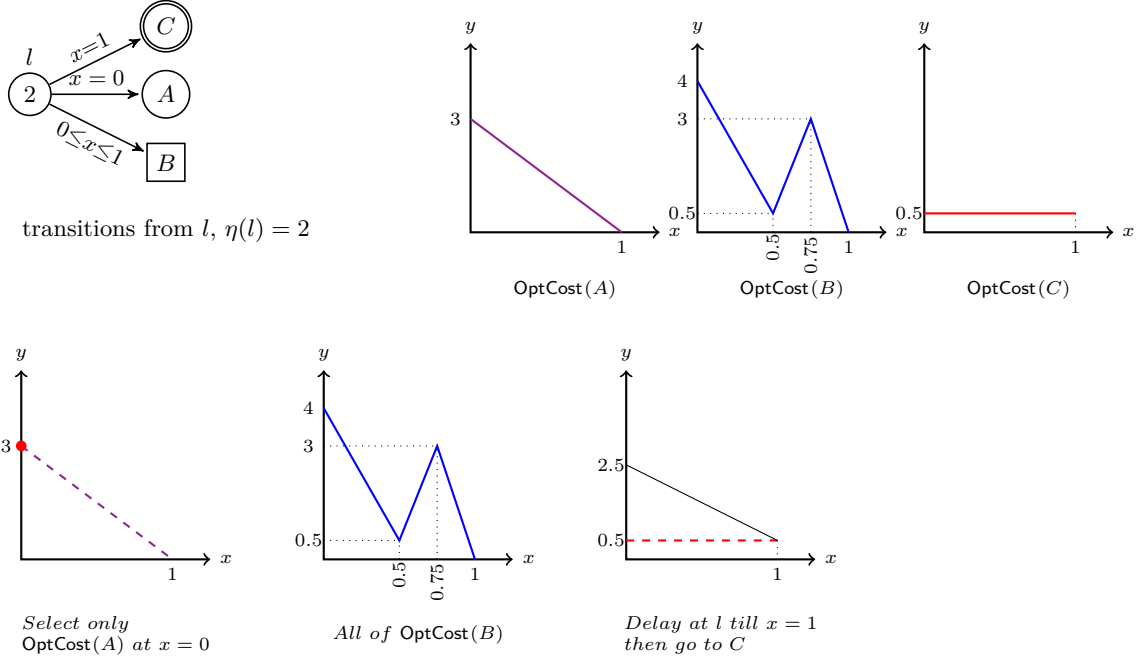
In transformation 1 from RPTG to dwell-time PTG, we replaced the constraint $H < x < H+1$ by $H - \delta < x < H + 1 - \delta$. Such a constraint would correspond in the resetfree FRPTG to $1 - \delta < x < 1$ or $0 \leq x < 1 - \delta$. We have already dealt with the constraint $0 < x < 1 - \delta$. Now, we shall highlight the difference to make it work for $1 - \delta < x < 1$. We shall compute as usual, by applying the steps of Algorithm 1, and also get the prescribed strategy out of the final function f' . Now if the computed strategy σ for l suggests to take a transition in the interval $[0, 1 - \delta]$ then instead of this transition we prescribe waiting at l . This is because the guard on the outgoing transition(s) is $1 - \delta < x < 1$. The rest of the strategy prescribed by f' over $(1 - \delta, 1]$ is retained as is.

Constraints on outgoing edges are $x = 0, 0 \leq x \leq 1, x = 1$

Figure 15 explains how to solve for optcost if the outgoing transitions have different guards.

As Player 1 can go to A only if $x = 0$, we need to consider only that point of $\text{OptCost}(A)$ while computing the optcost of l . Similarly, player 1 can go to C only when $x = 1$. Thus the function to consider, for taking the transition to C is the cost of the path (or action) of delaying in l till $0 \leq x < 1$ and going to C at $x = 1$. Upon reaching C at $x = 1$, the cost incurred will be $\text{OptCost}(C, 1) = 0.5$. Delaying at l at the rate of $\eta(l) = 2$ yields a function with slope -2 passing through the point $(1, 0.5)$ (corresponding to going to C at $x = 1$).

⁴ If there are n transitions in the longest path from source to target then $t = \frac{\epsilon}{n * (m_{max} + 1)}$.



■ **Figure 15** Optcost Computation for different guards

H l_{min} is a player 2 location

Here we discuss in detail how to take care of the dwell-time requirements while running Algorithm 1. Recall that there are three kinds of player 2 locations : urgent, those with dwell-time requirements $[0, \delta]$ and those with dwell-time requirements $[\delta, 2\delta]$.

1. **Urgent location** : superimpose and take exterior (only Steps 1 and 2).
2. **$[0, \delta]$ -delay location** : From Lemma 15, we know that Player 2 will want to spend as much time as possible at a location l_{min} while keeping $x \leq v_i$ whenever there is a function g_i over $[u_i, v_i]$ whose slope is $> -\eta(l_{min})$. Note that we proved Lemma 15 for player 1, however, an analogous result works when l_{min} is a player 2 location.

Thus, if l_{min} is entered at $x = \nu \in [u_i, v_i - \delta]$, then player 2 spends δ time and exits (as δ is the maximum delay permitted in l_{min} by the dwell-time restriction) at $\nu + \delta$ to the successor as prescribed by f at $x = \nu + \delta$. If l_{min} is entered at $x = \nu \in [v_i - \delta, v_i]$, then player 2 spends $v_i - \nu$ at l_{min} and exits at v_i to the successor as prescribed by f at v_i . In the superimposed optcost function f , a function $g_i : y = -mx + c$ having domain $[u_i, v_i]$ with slope less than $-\eta(l)$ is replaced as follows : alter g_i from $y = -mx + c$ to $y = -m(x + \delta) + c + \eta(l) * \delta = -mx + c + (\eta(l) - m) * \delta$ for $x \in [u_i, v_i - \delta]$. Let us denote the new function as h_i over the domain $[u_i, v_i - \delta]$. This corresponds to spending δ time until $x \leq v_i$.

When $x \in [v_i - \delta, v_i]$, then Player 2 spends the time $v_i - x$ at l_{min} before proceeding, as prescribed by f from v_i onwards. Thus the function obtained by replacing g_i for this range $[v_i - \delta, v_i]$, h'_i is $y = -\eta(l_{min})x + c'$, and passes through the point $(v_i, f(v_i))$. However, h'_i should intersect with h_i at $v_i - \delta$ to make the resulting improved optcost function continuous (and thus usable by the predecessors of l_{min}). We shall show that the line passing through the two points $(v_i - \delta, h_i(v_i - \delta))$ and $(v_i, f(v_i))$ has a slope $-m' = -\eta(l)$. We have g_i , the original cost function, and h_i , and we know that from v_i onwards, Player 2 has to continue with the optcost as dictated by f (the superimposed function). Thus we know that from the point $(v_i - \delta, h_i(v_i - \delta))$ of the new function h_i , the optcost will proceed

towards the point $(v_i, f(v_i))$ (recall that $g_i(v_i) = f(v_i)$). Thus given these two points, we find the line $h'_i = -m'x + c'$ as follows.

$$\begin{aligned}
 -m' &= \frac{f(v_i) - h_i(v_i - \delta)}{v_i - (v_i - \delta)} \\
 &= \frac{[-m * v_i + c] - [-m(v_i - \delta) + c + (\eta(l) - m) * \delta]}{\delta} \\
 &= \frac{-\eta(l) * \delta}{\delta} \\
 &= -\eta(l)
 \end{aligned}$$

Similarly, we also find c' by using $h'_i = -m'x + c'$ where slope is $-m' = -\eta(l)$ and this line passes through the point $(v_i, f(v_i))$.

$$\begin{aligned}
 f(v_i) &= -\eta(l) * v_i + c' \\
 -m * v_i + c &= -\eta(l) * v_i + c' \\
 c' &= c + (\eta(l) - m) * v_i
 \end{aligned}$$

3. **$[\delta, 2\delta]$ -delay location** : For every function g_i in f (the superimposed function) of Step 2, we first apply the modification of always spending δ delay at l_{min} . This is achieved by changing it from $y = -mx + c$ to $y = -m(x + \delta) + c + \eta(l) * \delta$. The domain of g_i also changes from $[u_i, v_i]$ to $[u_i - \delta, v_i - \delta]$. Thus the entire superimposed function f has been modified to f' (lets call it the adjusted superimposed function). After this, proceed with l_{min} as though it were a $[0, \delta]$ -delay location while taking f' to be its adjusted superimposed function.

H.1 Complexity and Termination when l_{min} is a player 2 location

Computing Almost Optimal Strategies: The strategy corresponding to computed optcost when l_{min} is a player 2 location is derived as follows.

1. l_{min} is urgent. In this case, we simply do steps 1,2 of the algorithm, superimpose and take exterior obtaining the function f . For $x \in [u_k, v_k]$, the strategy will dictate moving to location l_k , since g_k is the optcost function over the domain $[u_k, v_k]$ of the successor l_k of l_{min} .
2. l_{min} is a $[0, \delta]$ -dwell time location. If $x \in [u_i, v_i - \delta]$ and the function is h_i , the strategy will prescribe waiting at l_{min} for δ amount of time and then proceed to l_i whose cost function is g_i , the one replaced by h_i . If $x \in [u_i, v_i - \delta]$ and the function is g_i (not replaced at Step 3), then the strategy suggests going immediately to l_i whose cost function is g_i . Finally, if $x \in [v_i - \delta, v_i]$ for functions h'_i , we prescribe waiting at l_{min} till $v_i - x$.
3. l_{min} is a $[\delta, 2\delta]$ -dwell time location. The strategy prescribes waiting for δ time at l_{min} , and then uses the strategy prescribed above for $[0, \delta]$ -dwell time locations.

We have already discussed the complexity of Algorithm 1 in computing the optcost function for l_{min} , and the almost optimal strategies when l_{min} is a player 1 location. Now we discuss the case when l_{min} is a player 2 location.

Assume l_{min} is a player 2 location. Let $\alpha(m, p)$ denote the total number of affine segments appearing in cost functions across all locations. We handle this case by making l_{min} urgent and solve the modified PTG \mathcal{G}' (where l_{min} is urgent) which has one location less and then uses the computed optcost functions as outside cost functions to solve for l_{min} itself. This can be repeated as the optcost computed in \mathcal{G}' could get updated when the optcost cost of l_{min} itself is computed. This process gets repeated as many times as the number of segments we started with i.e p . Thus the equation is $\alpha(m, p) \leq p \cdot (1 + \alpha(m - 1, p + 1))$ where $\alpha(m - 1, p + 1)$ is the number of segments used for solving \mathcal{G}' . $1 + \alpha(m - 1, p + 1)$ is the number of segments

used for solving for l_{min} and $p(1 + \alpha(m - 1, p + 1))$ are the repetitions. Solving this, one can easily check that $\alpha(m, p)$ is at most triply exponential in the number of locations m of the resetfree component $\mathcal{G}_{\mathcal{F}}$. Obtaining a bound of the number of affine segments, it is easy to see that Algorithm 1 terminates; the time taken to compute almost optimal strategies and optcost functions is triply exponential.